An Integrated Architecture for Learning, Planning and Reacting Applied to a Real Mobile Robot

Alex Weiser Pedro Lima

19 de Setembro de 1995

RT-401-95

ISR-Torre Norte
Av. Rovisco Pais
1096 Lisboa CODEX
PORTUGAL

Relatório do trabalho final de curso efectuado no ISR pelo aluno finalista da Technische Universität München, Alex Weiser.

Intercâmbio apoiado pelo programa ERASMUS.

Keywords: Reinforcement Learning, Planning, Reactive Planning, Mobile Robotics

Abstract

The experiment performed here consists of the development of an algorithm in which a mobile robot located in a completely unknown environment should find the goal and, in a sequence of trials, establish the shortest way to reach it. Based on a algorithm in which the robot learns from trial and error in an incremental way, it builds a world model and uses that model to plan reactions that result in an optimal action sequence to reach the goal.
The world model tries to resemble the environment (real world) by dividing the area into square cells. From each cell the robot can execute four actions to reach one of the neighbouring cells. All cells in the world model are initially assumed as being unknown. After trying to reach an unknown cell it will be set as being occupied if there is an obstacle (the robot then goes back to the previous cell) or set as empty if the action was successful. A policy decides what action is going to be taken and an evaluation function evaluates each cell in order to bring the robot to the goal. Both policy and evaluation are updated iteratively after each step depending on a reward received from the world by reaching the new cell after the step has been executed, and by a temporal difference learning method.
In one experiment the robot executes a sequence of trials trying to find the best path. A trial consists of a sequence of real and hypothetical steps starting from the initial position to the goal. A real step is made when the robot tries to move into another cell. After each real step a fixed number of hypothetical steps is executed in which the robot doesn't move but executes a learning and planning step in order to improve the policy. In a hypothetical step, the algorithm uses the world model instead of the world and chooses the position in which to execute the step randomly.
The algorithm shows good results in the learning process after a relatively long first trial. The speed of learning depends on the number of hypothetical steps after each real step and the rate of modification of the policy.
The algorithm was simulated and tested by developing a software module in C language and implemented in a mobile robot platform with sonar scanning for obstacle detection and dead reckoning for localization.
This implementation intends to verify the concept, and shows its functionality by using an internal model built from real sensors to speed up the learning processes.

I would like to thank Prof. Pedro Lima and also João Sequeira for their neverending support and all the nice colleagues from Instituto de Sistemas e Robótica.

I would like to thank Univ.-Prof. Dr.-Ing Georg Färber and Alexa Hauck from Lehrstuhl für Prozeßrechner for their willingness to support my work on my diploma thesis at the Instituto Superior Técnico.

Contents

List of Figures

List of Tables

Symbols and Abbreviations

#	represents an obstacle in a map
a	action
α	rate of updating of the policy
AI	artificial intelligence
$ASDP$	albatros software developing system
β	rate of updating of the evaluation
com	command
CPU	central processing unit
E	expected value
e	evaluation
e'	posterior evaluation
$Eval$	evaluation function
γ	factor used to calculate the posterior evaluation
G	goal
I/O	input/output
LAN	local area network
PID	proportional integral derivative
r	reward
S	start position
US	ultrasonic
x	actual state or position
y	state to be reached

Chapter 1

Introduction

To solve a job or task a mobile robot in general has to determine a path. The robot normally knows its own position and the postition of the goal. The problem is in finding the shortest and still safe way from the start position to the goal without taking the system into a barrier or obstacle.

In traditional methods used in artificial intelligence the system deduces its best action in light of its current goals and world model. However, it is widely recognized that those methods usefulness is limited by its computational complexity and by its dependence on a complete and accurate world model.

An alternative approach is to plan in advance and compile the results into a set of rapid reactions or situation-action rules. These rules are then used for real time decision making.

A third approach is to learn reactions by trial and error. This has the advantage of eliminating the dependence on a world model.

The algorithm developed in this paper is based on the so called Dyna System[Sutt 90], and integrates the three above mentioned approaches. The architecture learns from trial and error while it builds a world model and uses that model to plan reactions that result in optimal action sequences.

The algorithm was first tested in experiments using a simulated world and then implemented in a mobile robot using dead reckoning for localization and sonar sensoring for obstacle detection.

Chapter 2

The Dyna Algorithm

2.1 Basic Idea

The Dyna algorithm is based on the idea that planning is like a trial-and error learning from hypothetical experiences.

The algorithm consists of four primary components as shown in figure 2.1:

Policy Is a function formed by the current set of reactions. It receives as input a description of the current state[1] in the world and produces as output an action to be executed. The reward is used in the learning process to update the policy.

World Represents the robot's external environment. In general the world provides the reward due to a real action, e.g. movement of the robot. The world is also designated as the real world, throughout this paper.

World Model A world model is included to mimic the one-step input-output behavior of the real world. This model is updated in the real experiences and used in place of the world in the hypothetical experiences.

Evaluation Function Evaluates each state known in the world model providing a parameter used to update the policy.

According to the information contained in the policy, the algorithm chooses an action that leads to a new state in the world. The reward given by the world for reaching that new state is a parameter to update the evaluation. The higher the difference between old and updated evaluation, the stronger the policy is modified in order to increase or decrease the probability to do the same action again. A world model is built with the information acquired by the interaction with the real world, and is used to perform the hypothetical experiences.

The evaluation function, the policy, and the world model are each updated by separate learning processes. The evaluation of a state should be equal or proportional

[1]The expression *state* is used instead of *position* throughout this paper.

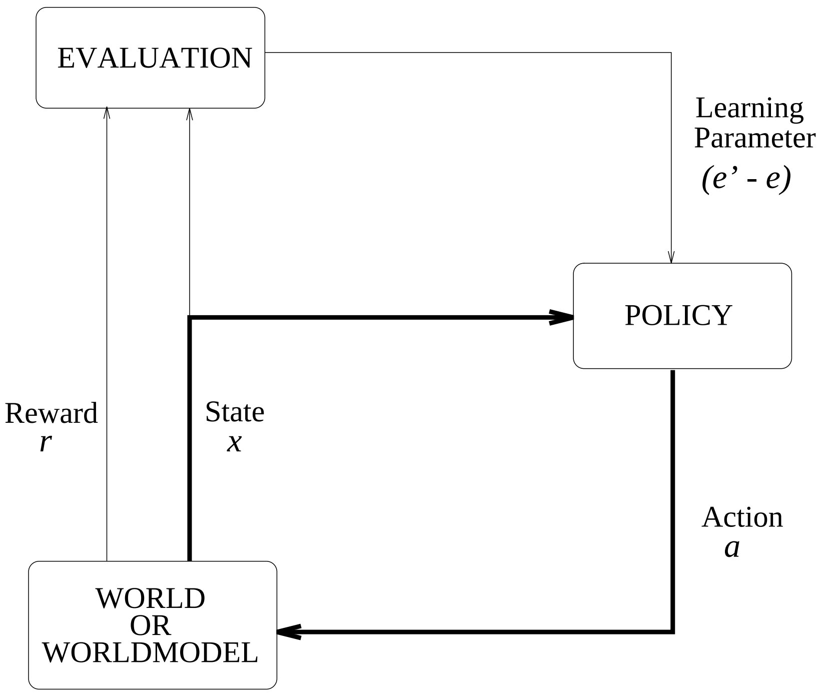


Figure 2.1: Overview of Dyna

to the evaluation of the state with the highest evaluation that can be reached in one action, taking into consideration the reward (or cost) r for that one transition.

$$Eval(x) \text{``}=\text{''} \max_{a \in \text{actions}} E\{r + Eval(y) \mid x, a\} \tag{2.1}$$

where $E\{\ \cdot \mid \cdot\ \}$ denotes a conditional expected value. The equal sign is quoted to indicate that this is a condition we would like to hold, not one that necessarily does hold because there is no complete world model.

2.1.1 The Learning Process

For a fixed policy the algorithm works like a simple reactive system, see Figure 2.1 considering only the thick lines. However, the policy is continually adjusted by an integrated planning and learning process that can be accelerated by generating any number of training examples. These training examples represent the hypothetical experiences.

If the learning process converges such that the Equation (2.1) holds in all states, then the optimal policy for each state is given by the action a among the set of actions that results in the maximum value on the right hand side of the equation.

2.2 Outline

The description of the algorithm in Section 2.1 gives an general idea about relaxation planning. The method used here is based on a iterative updating of the policy, in which the evaluation function and policy are simultaneously approximated. In addition Dyna is a Monte Carlo approximation variant in which the world model need only be sampled, and not examined directly. That makes it easy to use the world model or real world alternatively in this method.

A very simple architecture is used. The same algorithm is applied to both real steps and hypothetical steps[2]. While in a real step the world is used, the hypothetical steps are executed in the world model. The results of both cases are accumulated in the policy and evaluation function.

[2]An experience consists of a set of steps.

An outline of the algorithm follows:

1. Decide if it is a real step or a hypothetical one.
2. Pick a state x. If this is a real step, use the current state. If this is a hypothetical experience, choose a random state that is already in the world model.
3. Form prior evaluation of state x according to the result of the evaluation function: $e = Eval(x)$.
4. Choose an action a to be taken from state x by consulting the policy.
5. Execute the action a if there is no obstacle in this direction in the world and world model. If the action is executable, obtain next state y and reward r_y from world or world model.
6. Form posterior evaluation e' of the old state x using the evaluation of the new state y now reached: $e' = r_y + \gamma Eval(y)$.
7. If this is a real step, update x and r_y in the world model.
8. Update the evaluation function so that the evaluation of the old state x is closer to the posterior evaluation e' rather than to the prior evaluation e.
9. Update the policy. Strengthen or weaken the tendency to perform the action a from state x again according to the difference between the posterior and prior evaluation: $e' - e$.
10. Go to step 1.

If the world model is empty then it obviously doesn't make sense to perform hypothetical steps since an hypothetical action must always bring the system into a known state in the world model.

If there was an obstacle in the direction of action a then the systems returns or stays in the old state x and no updates of evaluation or policy are made.

Chapter 3

The Robot

3.1 Hardware Description

The robot, shown in Figure 3.1, consists of a mobile rectangular platform with the dimensions of 1000x700mm, built by the French company Robosoft. It has two independent propulsive wheels at the back and two free rotating and free turning wheels in the front.

The vehicle's operating system, Albatros, runs on a master CPU. A CPU for the ultrasonic subsystem is connected to the master CPU via VME-bus. The master CPU is a high performance double Eurocard CPU, compatible with VME-bus (IEEE1014 Revision CI). It is designed around a 32-bit 68020 microprocessor with 16 MHz clock. The board has 4 MBytes of dynamic RAM (accessible either from the VME-bus or from the 68020 processor), four JEDEC sockets taking Reproms up to 8Mb, four serial ports, a timer, and the 68882 floating point coprocessor.

3.1.1 Ultrasonic Subsystem

There are 24 ultrasonic transducers positioned around the vehicle for obstacle detection. These are interconnected by a LAUN[1] bus.

The unit installed in the platform is the Polaroid Ultrasonic Ranging Unit. It is capable of detecting the presence and distance of objects within a range of approximately 0.9 feet to 35 feet. The principal component is the transducer. It acts as both loudspeaker and microphone. A pulse is transmitted toward a target and the resulting echo is detected.

When a sensor is activated, it returns the distance of the first encountered object. The sensors are organized in 6 nodes with 4 sensors each. This allows the simultaneous activation of all the sensors of one node. The sensors connected to one node point to different directions so that interference between them is avoided. A VME-bus based

[1]The Local Area Ultrasonic Network(LAUN) is a measurement machine, designed for up to 90 ultrasonic sensors connected through a common bus.

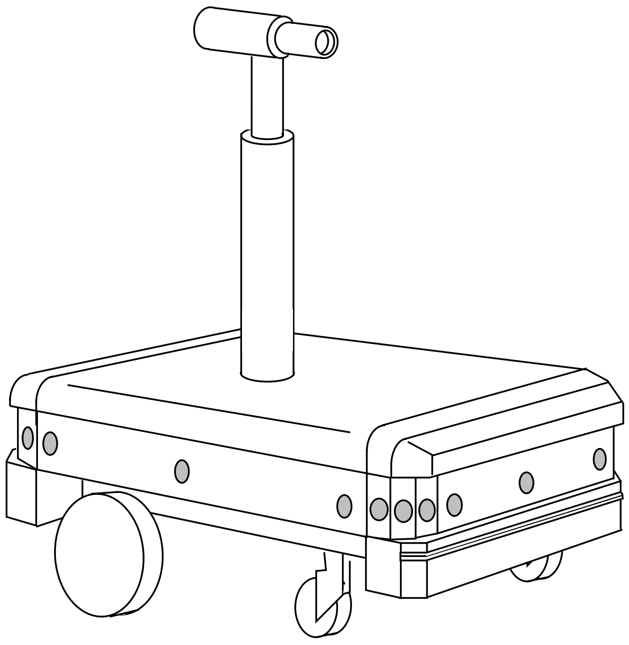


Figure 3.1: Robuter Platform

numerical controller (US CPU) is used to control the ultrasonic bus and to perform the low-level processing, it is connected to the vehicle's master CPU by the VME-bus.

The US CPU can use two modes of reading sensor values. In the first mode, it waits for the master CPU to ask a new value, reads the value and then waits for a new request. In the second mode, the US CPU continuously updates a sensor value table. This table is also accessed by the master CPU, so that a read request only results in retrieving the sensor values in the table rather than actually performing a sensor read.

The US CPU is a double-high VME-module with high performance processing designed to function as standalone microcomputer, as a single CPU-controler in a VME-bus system, or as a single element in a multiprocessor VME-bus configuration. This module features Motorola MC68000 16-bit microcomputer with address range of 16Mb. Sockets are provided for up to 256Kb of user-supplied memory. Synchronous and asynchronous serial communication at up to 19.2 baud is supported through two front panel ports and two independent 8-bit parallel communication channels are available.

3.1.2 Odometric Subsystem

The odometric system is based on a circular approximation and uses extended precision floating-point arithmetics. This approach provides a high accuracy but is time consuming in return. The optical encoder counts 60 points per revolution.

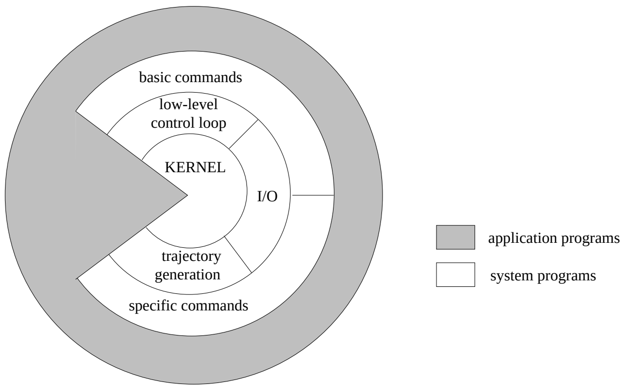


Figure 3.2: Software Organization of an Albatros System

3.2 The Albatros Operating System

Albatros is a real-time operating system specifically designed to control, with minimal hardware and maximum efficiency, any kind of multiaxis and multisensor device. It has a multi-tasking organization with predefined priorities and fixed scheduling for tasks.

The whole system is divided into two sets of separate programs:

system programs which is the resident part of Albatros (firmware). They contain the basic primitives wich allow the user to access the lower levels of the system. They include the real time kernel, the I/O drivers, the generalized PID, trajectory generator and sensor read modules, and the command interpreter.

application programs which depend on the application and are developed by the user in a high-level language on a host machine.

The application programs can communicate with the Albatros system programs through parameterized ASCII messages that are processed by the command interpreter or by system calls. The ASCII commands can be sent from a host machine through a 9600 baud serial line or from any application program by using the $send(com)$ function where the parameter com is the same ASCII command.

The application program can share the same processor with the Operating System allowing to use its resources through system calls. It is possible to link the Albatros kernel with the user's specific programs. The figure 3.2 shows the typical organization of an Albatros system.

3.2.1 Developing Applications

The general principle to develop applications for an Albatros machine is to write and edit the source code on a host machine, compile it and include the references to Albatros at link time. The object modules of the developed program are linked with standard

Albatros access libraries and converted into S record format, a format defined by Motorola for transfer of executable code, and then downloaded in the Albatros machine. As shown in figure 3.3.

A set of access libraries provide a C language interface to the monitor. The use of the access libraries is very useful when implementing time-critical applications because they are more efficient than the *send*() command since they don't involve the command interpreter.

The Albatros package provides five different libraries:

- A library **libcstd** with most of the standard input/output routines (*gets*, *puts*, *printf* ...), string manipulation routines (*strlen*, *strcpy*, *strcat* ...) and mathematical functions (*sqrt*, *log* ...).
- The **libcalbl** contains low-level routines to access the device database and functions for sending any Albatros command to the command interpreter (*send*, *sendf*).
- The **libcalbh** library provides high-level routines for controlling device motions.
- The **libio** library contains general purpose input-output routines for serial ports.
- The **libcpro** library provides the primitives needed for the management of interruptible processes.

Since not all Albatros primitives have a correspondent C function, some ASCII has to be issued through the *send*() command. On the other hand, there are functions in the libraries that are not provided in the ASCII command list.

3.2.2 Process Management

In the version 4.0 of Albatros kernel, there are three types of processes that correspond to three different strategies of real-time process management. The type of processes are:

- system processes
- user uninterruptible processes
- user interruptible processes

The real-time process management performs the following actions at each sampling period:

1. interruption, generated by the real-time clock, whose time period is 10ms;
2. activation of the system process manager, which sequentially runs all the active system processes;

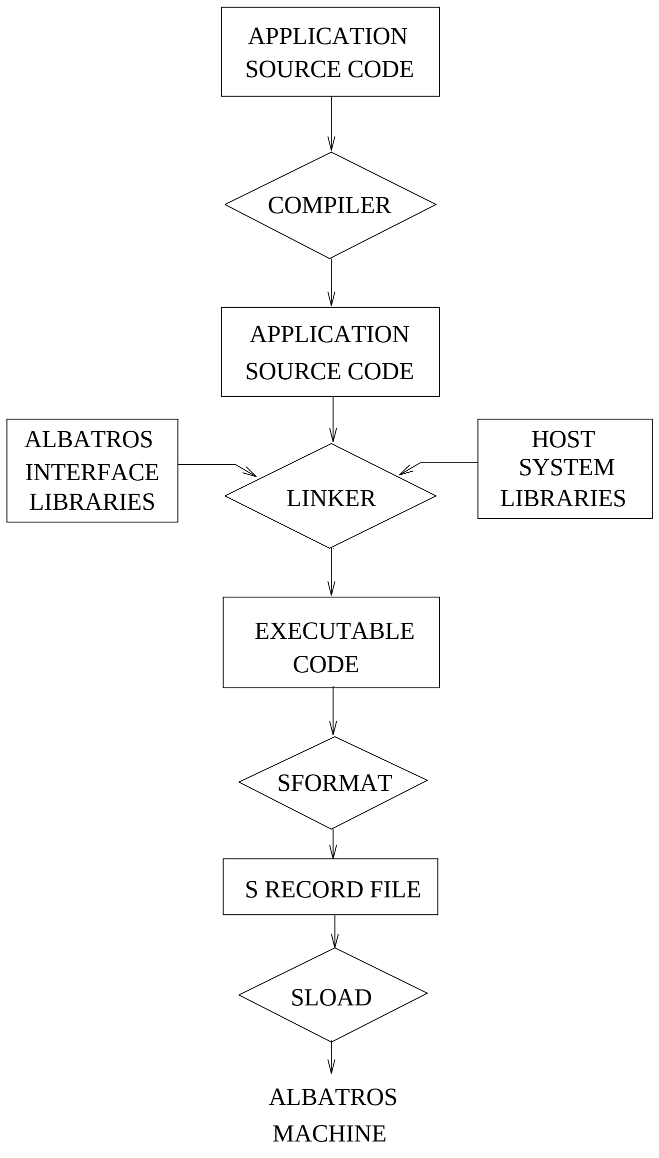


Figure 3.3: Albatros Software Development Package

3. activation of the user uninterruptible process manager which restarts the current active process;
4. activation of the user interruptible process manager which decides which process must be run, depending on the current time as well as the previous state of the user processes.

While system processes are generated by the system programs and cannot be accessed by the user, the other two types can be controlled by an application program. However, user uninterruptible processes can generate conflicts or deadlocks if they are too time consuming, and therefore not used here in application programs. This type of process management is obsolete and had been kept for software compatibility reasons due to older releases of the kernel.

User Interruptible Process Manager

The most important feature of the user interruptible processes is their ability to interrupt each other. In addition is it possible to launch several processes at the same time from the main process.

A priority can be associated to each process. A process with a high priority can interrupt a lower priority process, if its sampling period is elapsed. The main process has the lowest priority.

The system processes have the highest priority, and are run before all other existing processes. While all system processes must run within one time period, the other processes can take several periods to complete.

Chapter 4

The Task

4.1 Task Model

The model discussed here involves only the algorithm for learning and planning. It is executed under simulation of a real world. The implementation of a real robot in a real environment is a simple continuation of the model. It implies in additional modules for movement control and sensor reading and will be discussed later.

4.1.1 World Model

The space is represented by a grid of square cells, the dimension of a cell is dependent upon the vehicle to be used to execute the task and will be discussed later. For the simulation of the algorithm the size is not important and will be assumed to be 1.

The world model, therefore, consists of a matrix of cells. Each cell corresponds to a square shaped area in the real world. Status information about its reachability is maintained. The status can be on of the following: UNKNOWN, EMPTY or OCCUPIED. As in Figure 4.1.

If the robot has already been in a cell, it contains information about the reward and

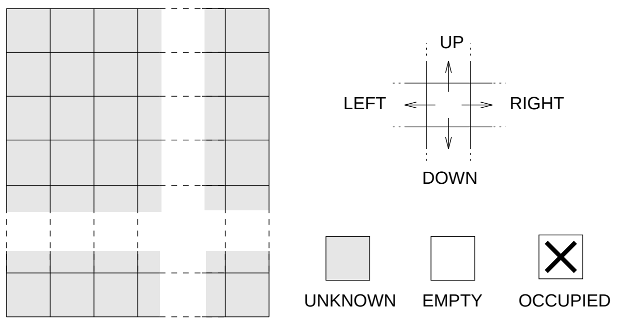


Figure 4.1: The World Model

whether this cell can be reached or not without colliding with an obstacle. From each cell there are four possible actions that can be taken to reach one of the four neighboring cells. The actions are: UP, DOWN, LEFT and RIGHT. An action changes the state accordingly, except where such a movement would take the system into a barrier. The barriers are introduced in the model by simulating an alarm that later on will be activated by the sonar sensors, when the system tries to execute a forbidden action.

The reward is zero for all transitions except for those that lead to the goal state, for which it is +1.

4.1.2 Policy and Evaluation Functions

The evaluation function, the policy and the world model were implemented as lookup tables.

The policy has an entry w_{xa} for every pair of state x and action a, that means four entries for each state in the world model. Actions are taken stochastically by turning the four policy entries for the actions in probabilities using a Boltzmann distribution. The probability for each action a from state x is then represented by $P\{a \mid x\}$ according to Equation(4.1):

$$P\{a \mid x\} = \frac{e^{w_{xa}}}{\sum_j^{\text{actions}} e^{w_{xj}}} \tag{4.1}$$

The Equation (4.1) forces the sum over the probabilities of the directions to be equal to 1. A decision for the direction to be taken is made by using a random number between zero and one. As shown in the left diagram of Figure 4.2 when the policy table is initialized, the probability for each direction is 0.25. Thus the height difference between a column and the prior one is the same (shaded area). While the width of the column is constant and represents the action to be taken, the height varies depending on the probability of the direction it represents. When an update occurs indicating that the action *down* leads faster to the goal, the diagram may look like the right side of Figure 4.2. The chaded area representing the probability for the action *down* increased and consequently it is more probably for an random number to be in this range. The height of the columns changed in order decrease the probability for the other actions.

The evaluation table is updated according to the simplest method of temporal difference learning as shown in Equation (4.2). The difference between the post and prior evaluation is multiplied by a factor β and than added to the last table entry that is equal to the prior evaluation. The factor β must be kept low. It represents the rate of approximation to a steady value. A high β can cause an oscillation around it.

$$Eval(x) = Eval(x) + \beta(e' - e) \tag{4.2}$$

The steady value of the state evaluation in direction opposite to the goal decreases by a rate of γ. The factor γ is used to calculate the posterior evaluation as shown in step 6 of the architectural outline at Section 2.2. The value of γ must be close to 1 so that the evaluation decreases smoothly, in this paper $\gamma = 0.9$ has shown acceptable performance.

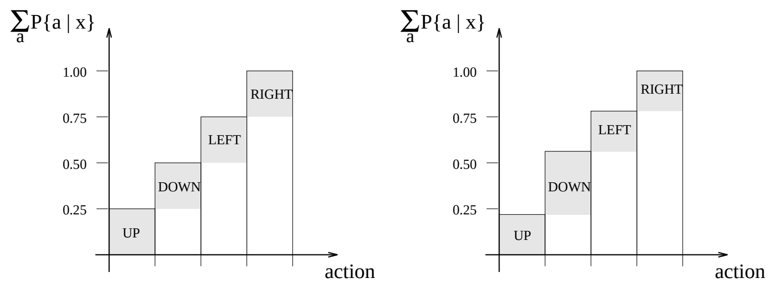


Figure 4.2: Probability for each action

The policy is updated in a similar way. The difference between post and prior evaluation is now multiplied with a factor α, that is a higher value situated between 100 and 1000, and then added to the last table entry.

$$w_{xa} = w_{xa} + \alpha(e' - e) \tag{4.3}$$

The parameter α is related to the rate of updating of the policy. The higher the factor, the faster the system will learn. The probabilities for each direction change by a higher value. If the value is too low a slow learning rate is achieved. If it is too high then the system goes with less probability to a still unknown state.

4.1.3 Example

The initial values in the evaluation table are zero, the world model is empty (all states are unknown except the start state) and the probability for each direction according to the evaluation table is 25%, as represented graphically in the chart of Figure 4.3. The number on the lower left corner of each cell indicates the evaluation and the sets indicate the policy (probability to perform each action). After a few trials, a policy and evaluation has been developed for almost every state, including the localization of obstacles and the goal state. An example of adjusted values is shown on the chart of Figure 4.4.

4.2 Real and Hypothetical Steps

The algorithm outlined before learns and plans from real steps and hypothetical experiences. A trial consists of a sequence of steps in the real world that brings the system from the start state to the goal. Upon entering the goal state, the system is instantly transported to the start state to begin the next trial. After each real step a fixed number of hypothetical steps is performed on the world model. In a hypothetical step states are selected at random uniformly over all states previously encountered. From

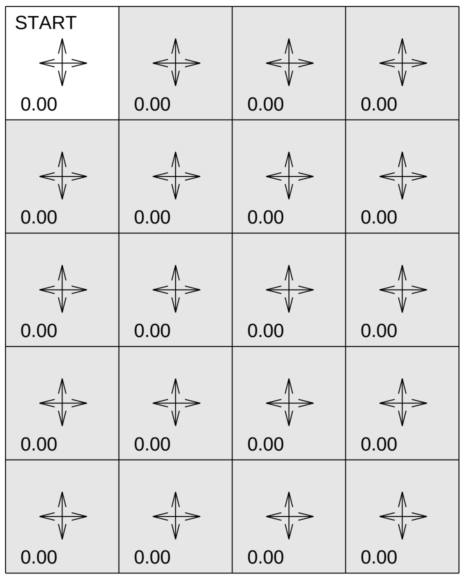


Figure 4.3: Initial values

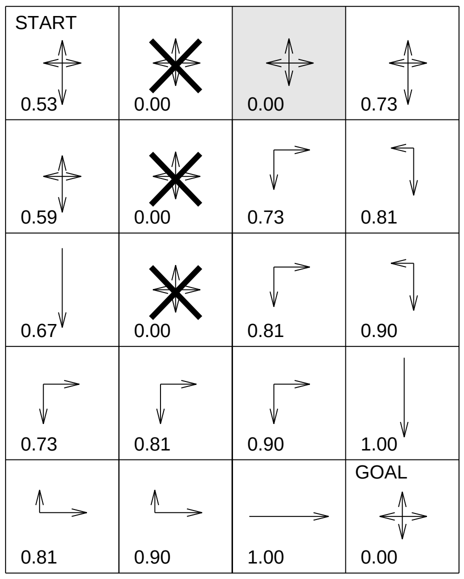


Figure 4.4: Example with adjusted values

this state an action among those that bring the system to a previously encountered state is also randomly selected.

In a hypothetical step the same sequence of actions is executed like in a real step with exception of the world model update. After an action has been executed a new state is chosen randomly instead of taking the last state reached.

Since in the first trial the evaluation table is empty the system does not learn, thus the hypothetical steps are omitted. It works as a reactive system and the policy keeps its initial values. This turns it into a random walk and the goal is found only by coincidence.

4.3 Limitations of the Model

The model used here limits the algorithm in many ways. The states and action spaces are small and enumerable, permitting tables to be used for all learning processes. This makes it feasible for the entire state space to be explicitly explored. For large state spaces it is not practical to use tables or to visit all states; instead one must represent a limited amount of experience compactly and generalize from it. The architecture is fully compatible with the use of a wide range of learning methods for doing this [Bart 83, Lin 94, Sin 92].

In this experiment, it was also assumed that the system has explicit knowledge of the world's state. In general, states cannot be known directly, but must be estimated with the pattern of past interaction with the world.

Another limitation of the algorithm presented here is the trivial form of search control used. It is simplified to the decision of whether to consider hypothetical or real experiences. The task considered here is so small that search control is unimportant.

Chapter 5

Simulation Experiments

5.1 Module DynAlg

The module `DynAlg` is a module developed in C language on a Sun Sparc workstation that executes the steps of the algorithm. It has an header file named `DynAlg.h` and the source code files are `DynAlg.c` and `Movements.c`. The listings can be found in Appendix A.

After starting the program the following parameters must be set by the user:

- starting point in the world model. It consists of the x and y coordinates and the mathematical positive angle in degrees relative to the x axis.
- maximum number of trials needed to avoid an infinite number of trials in the event the algorithm is not discontinued by the breakoff condition.
- number of hypothetical steps.
- seed number for the random number generator.
- goal state.

The functions $Initialize()$ and $SetParameters()$ perform the setting of system parameters. The size of the world model is defined by the defined constants in the header file `DynAlg.h`.

The module uses the functions $Rotate(Angle)$ and $Translate()$ to move from one state to another. They are implemented in the module `Movements.c`. During the simulation phase these two functions don't perform any action but return a boolean value indicating if the action has been executed or not. The $Rotate()$ function always returns `TRUE` while the $Translate()$ function can return `FALSE` if the state in which the translation ends allready occupied.

To prevent the system from leaving the perimeter of the world model an obstacle is simulated in every cell adjacent to it. These and all other barriers introduced in the simulation cause the $Translate()$ function to return `FALSE`.

	#			#	
#	16	#		3	#
#	9	#		2	#
#	5	#	1	6	#
#	7	4	4	3	#
#	7	12	1	1	#
	#	#	#	#	

Figure 5.1: Map of a world model with 42 cells

The algorithm performs only real steps in the first trial and introduces the hypothetical steps at the beginning of the second trial. If the number of steps needed to reach the goal is the same in two consecutive trials and also equal to the shortest number of steps needed among all the trials, then the optimal path is assumed to be found and the algorithm stops.

During execution, the states visited in each trial are saved in a form of map for each trial. The states and actions chosen in the hypothetical experiences are saved in a file. When the algorithm has been completed and the final policy table as well as the evaluation are also saved in files. The map is represented in table form as shown in Figure 5.1. The entries show how many times the robot has been in each cell. The sign # indicates that there is an obstacle in that cell. An empty entry indicates that the robot has not yet visited that cell.

The evaluation table is a matrix that contains an evaluation for each state and the policy table is a three dimensional matrix containing four entries for each state.

5.1.1 Predefined Constants

The most important predefined constants defined in the header file are:

XMAX – number of cells in the horizontal direction of the world model.

YMAX – number of cells in the vertical direction of the world model.

ACTIONS – number of actions that can be taken in a given state.

DX, DY – magnitude of one step. Equal to one because one action always takes the system to a neighbour cell.

STATEDIM – dimension of the state vector `pstate`.

X, Y, ANG – are the components of the state vector.

GAMMA – represents the value of 1000γ to calculate the posterior evaluation.

ALFA – represents the α to update the policy.

BETA – is the value of 1000β used to update the evaluation table.

5.1.2 Variable Types

In addition to the standard variable types in C, some types have been defined to facilitate the manipulation of variables. The definitions in the header file are:

pstate – pointer to an array of the type `State[STATEDIM]` that indicates the state in the world model.

boolean – boolean type that can be `FALSE` or `TRUE`.

switcher – similar to boolean and can assume the values `ON` or `OFF`.

action – actions type that can be `UP`, `DOWN`, `LEFT` or `RIGHT`.

infotype – type of information indicating the status of a cell in the world model. These are `UNKNOWN`, `EMPTY` or `OCCUPIED`.

5.1.3 Global Variables

The global variables defined in the header file are:

WorldModelCell[XMAX][YMAX] – is a matrix of struct variables. Each element contains the following structure:

Info – variable of type `infotype`.

Reward – integer containing the reward r for reaching the cell.

Frequency – integer containing the number of times the cell has been visited (this information is only used to build the map described in Section 5.1 at the beginning of this chapter and in Figure 5.1).

Eval[XMAX][YMAX] – matrix of double precision floating point numbers containing the evaluation for each cell. Corresponds to $Eval(x)$ in the algorithm description.

Policy[XMAX][YMAX][ACTIONS] – policy table. Each state or cell in the world model has one entry for each action. This corresponds to w_{xa} described in Chapter 4.

SonarAlarm – boolean value indicating an obstacle in direction of movement.

GoalFound – boolean that indicates if the goal point has been reached by the robot.

GoalPointSet – boolean that indicates that the goal has been located and its position is saved.

XGoal,YGoal – position of the goal.

The boolean variables concerning the goal make it easy to introduce a goal finding task. In the beginning the position of the goal is completely unknown and the variables `XGoal` and `YGoal` would have any invalid value. After localizing the goal, e.g. with an infrared camera, the position is saved and `GoalPointSet` is set to `TRUE` indicating that the variables `XGoal` and `YGoal` now contain valid information.

5.1.4 The Functions used in DynAlg.c

The functions used in the module `DynAlg.c` are the following:

RealStep(RealState) – routine that executes a real step on the state contained in the parameter as shown in the flow chart in Figure 5.2.

HypotStep(HypotState) – routine that executes a hypothetical step. See the flow chart in Figure 5.3.

Evaluate(State) – evaluates a state like the function $Eval(x)$ in the algorithm description. It consists of a reading from the evaluation table.

ChooseAction(State,pAction) – chooses the action to be taken from a state by consulting the policy using a Monte Carlo distribution. It returns `pAction` that is a pointer to a variable of type `action`.

GetNextState(State,Action,NewState) – function that consults the entries in the world model of the state that will be reached with action `Action` from state `State`. Provides a pointer of type `pstate` to the new state and returns the `Info` of that state.

DoRealAction(RealState,Action,NextState) – tries to execute the real action and returns a boolean action corresponding to the success of the action. It updates the `RealState` pointer accordingly. This routine provides an interface to the module `Movements.c` through the functions `Rotate()` and `Translate()` used later to move the real robot. Before trying to move, it invokes a function `GenerateSonarAlarm()` that can simulate an alarm for that action if there is supposed to be an obstacle.

FormPostEval(NewState,Reward) – calculates and returns the posterior evaluation like the e' in the algorithm description.

UpdateWorldModel(State,NextState,Info,ActionReward) – updates the values of `Info` and `Reward` in the state `NextState` of the world model and increments `Frequency` in `State` or `NextState` depending on `Info` which is related to the success of the movement and therefore to the actual position of the robot.

UpdatePolicyEval(State,PriorEval,PostEval,Action) – updates the evaluation of `State` and the policy of `Action` according to the difference between `PostEval` and `PriorEval`.

GetReward(State) – returns the reward for reaching the position in the world. The returned value is 1 if `State` points to the position `(XGoal, YGoal)` and 0 otherwise.

IsGoal(State) – returns `TRUE` if `State` is the goal and sets the boolean variables `GoalFound` and `GoalPointSet` if applicable.

Initialize() – initializes the evaluation table, the policy table and the world model.

SetParameters(pHypotCycles,pTrialMax,StartState) – prompts the user to enter parameters like number of hypothetical cycles, number of trials, `StartState` and `Goal` positions. It also prompts for the seed and initializes the random number generator.

MakeTables() – writes the the evaluation table and the policy table in a file.

ShowMap(Trial) – writes the information of the world model in a file after each trial.

main() – controls the flow of the algorithm as shown in Figure 5.4.

5.1.5 The Random Number Generator

The random number generator is used very frequently in this algorithm and has a large influence on the direction of movement and therefore the duration of a trial. This is especially true when the policy is still poor. It has an influence in every action chosen in real and in hypothetical steps. It is also used to choose the beginning state for every hypothetical step. Thus, it is important to describe it more precisely.

The functions used are $srand48()$ and $drand48()$. The first initializes the random number generator using a seed value given by the user. The seed is the value to begin randomization determining the randomization pattern. The second function returns nonnegative double-precision floating-point values uniformly distributed over the range from 0 to 1.0. It stores the last value into an internal buffer. For that reason it must be initialized prior to invocation.

Both functions work by generating a sequence of 48bit integer values X_i, according to the linear congruential formula in Equation (5.1):

$$X_{n+1} = (aX_n + c) \bmod m \quad \text{with } n \geq 0 \tag{5.1}$$

The parameter m equals 248, hence 48bit integer arithmetic is performed. The multiplier value a and the added value c are given by:

$$\begin{aligned} a &= 5DEECE66D16_{hex} = 2736731631558_{dec} \\ c &= B16_{hex} = 138_{dec} \end{aligned}$$

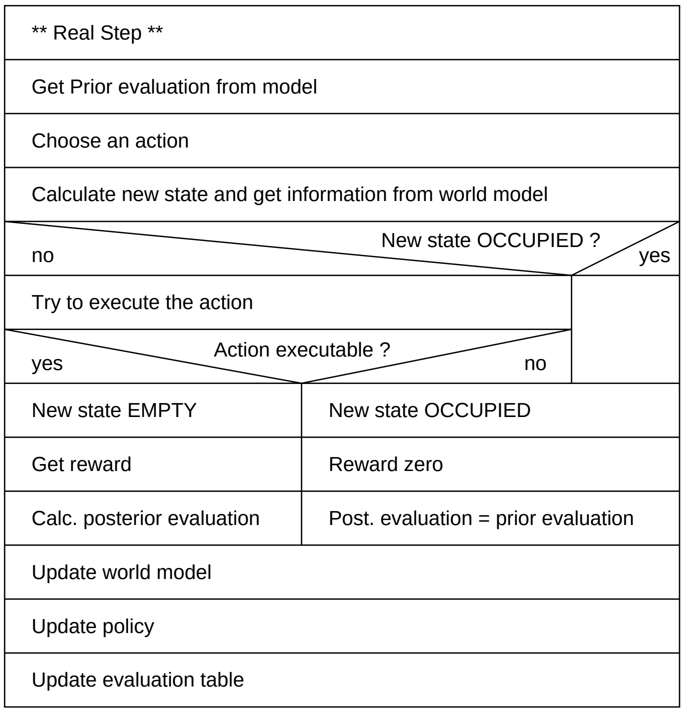


Figure 5.2: Flow Chart of a Real step

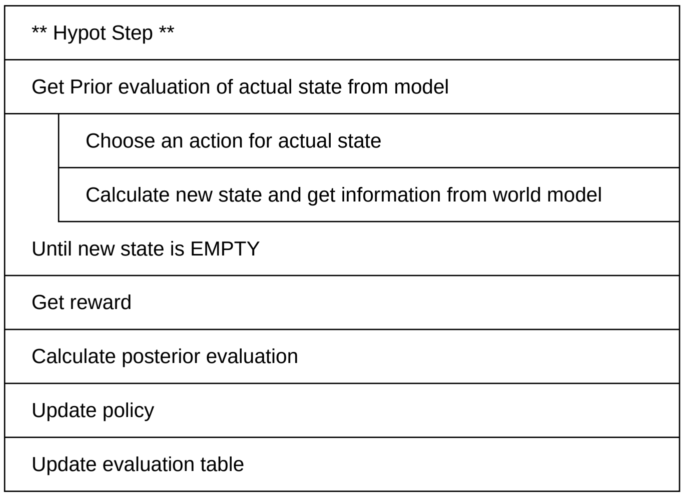
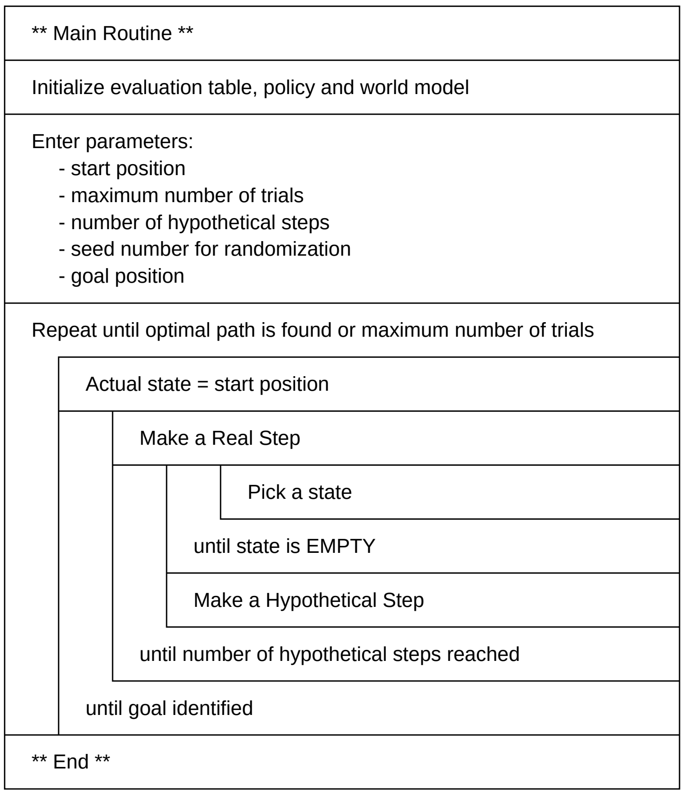

** Hypot Step **
Get Prior evaluation of actual state from model
Choose an action for actual state
Calculate new state and get information from world model
Until new state is EMPTY
Get reward
Calculate posterior evaluation
Update policy
Update evaluation table

Figure 5.3: Flow Chart of a Hypothetical step



```
** Main Routine **
Initialize evaluation table, policy and world model
Enter parameters:
    - start position
    - maximum number of trials
    - number of hypothetical steps
    - seed number for randomization
    - goal position
Repeat until optimal path is found or maximum number of trials
    Actual state = start position
        Make a Real Step
                Pick a state
            until state is EMPTY
            Make a Hypothetical Step
        until number of hypothetical steps reached
    until goal identified
** End **
```

Figure 5.4: Flow Chart of the Main routine

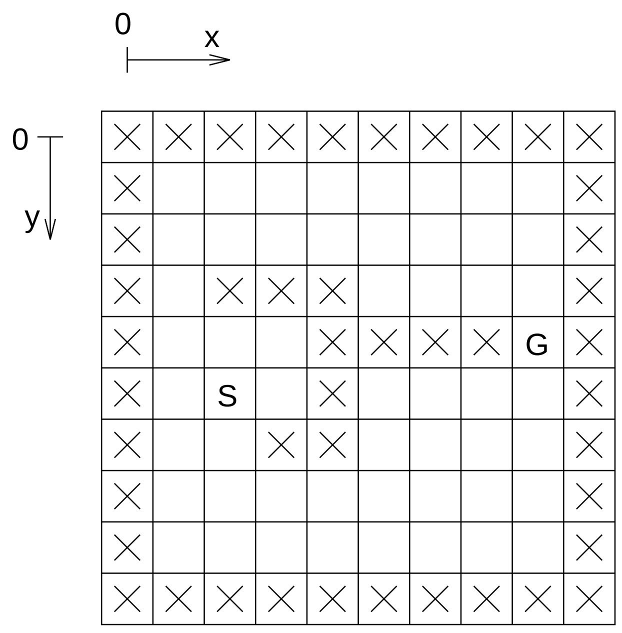


Figure 5.5: Position of the obstacles in the experiment.

The return values are computed by first generating the next 48bit X_i in the sequence. Then the appropriate bits are copied from the high-order (most significant) bits of X_i and transformed into the return value.

5.2 Results of the Simulation

A complete printout of all the results is provided in the Appendix B. Simulationd were made by varying the number of hypothetical steps, the seed numbers and the values of α.

In the simulation, the size of the world model used was 10 by 10 cells. Obstacles were introduced in the positions as shown in Figure 5.5. S indicates the start position and G the goal. This example is special in that it divides the world into two parts and there is a shortest path from each side to the goal. The path through the lower part is the best. The question now is whether the algorithm is able to choose the correct path.

In prior versions, the algorithm was used in a slightly different form. The hypothetical steps were performed only after the robot had reached the goal after a sequence of real steps, and not after each real step as is presently the case. This can be interpreted as a particular case from the experience executed now. The main difference is that the hypothetical steps are counted in a different way, and the number of hypothetical steps is independent of the number of real steps.

5.2.1 Learning Curve

Figure 5.6 shows the results of the simulation first with no hypothetical steps at all and then varying the number of them. The seed value was 1, and $\alpha = 1$. It is easy to observe that the curve falls more quickly the more hypothetical steps are performed, thus the system learns faster.

The resulting policy tables in Appendix B.2.1, B.2.2 and B.2.5 show that the probability for each direction have changed very fast, some of them have 100% probability making it impossible for the robot to choose another direction.

To slow the rate of change of the policy, another experiment was made using a lower value of α. Two curves show the performance of learning by two different values of α with 10 hypothetical steps in Figure 5.7 and 100 hypothetical steps in Figure 5.8. More interesting is to observe the difference between the policies shown in Appendix B.2.4 and B.2.7, even after finding the shortest path, the probabilities are not as high as they were before in the first experiment.

Finally another experiment was made using different seed values to initialize the random number generator to demonstrate its influence on the results. As shown in Figure 5.9 the number of trials needed is almost the same, but the robot performed completely different sequences of actions during the trials. Compare the maps in Appendix B.1.2 and B.1.3.

One can see in the path the robot chose in the Appendix B.1.6 that the experiment of Figure 5.10 using seed = 5 it chose first the upper part of the world and then the lower part. Sometimes it happened that the system was only one step away from the goal and then it chose a completely different direction. That happened on the first trial of the experiment in Figure 5.9 using seed = 2, as shown in Appendix B.1.3. Another interesting fact in that experiment is that although knowing almost all the states in the world it couldn't choose the shortest path to reach the goal, until the algorithm was discontinued by the breakoff condition.

5.3 Conclusions

Without hypothetical steps, each trial updates only the policy of one cell beginning with the last cell before entering the goal. This means that during the second trial the entries of the world modell have changed from UNKNOWN to OCCUPIED or EMPTY for every cell the robot tried to reach, but the policy of only one cell has been updated.

Using hypothetical steps, the first trial also updated only the policy of the last cell before entering the goal. During the second trial, however, an extensive policy has been developed that, by the trial's end, will reach almost back to the start state depending on the number of planning steps.

The higher the value of α the higher the rate of change of the policy and the lower the probability to reach a completely new, yet unknown, state. That could prevent the system from finding the optimal path.

The simplifications made in this experiment make the problem more like the kind

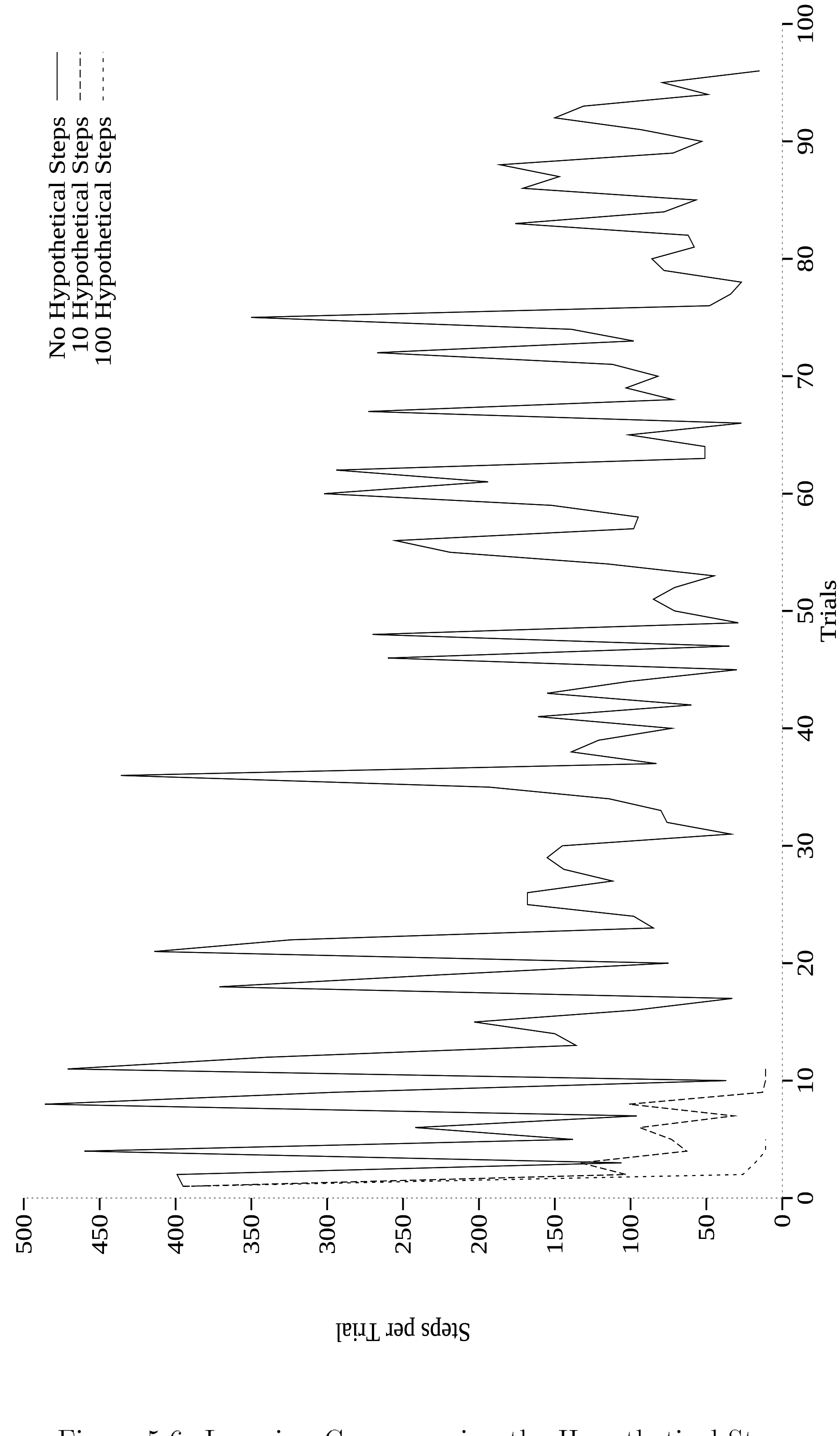


Figure 5.6: Learning Curve varying the Hypothetical Steps

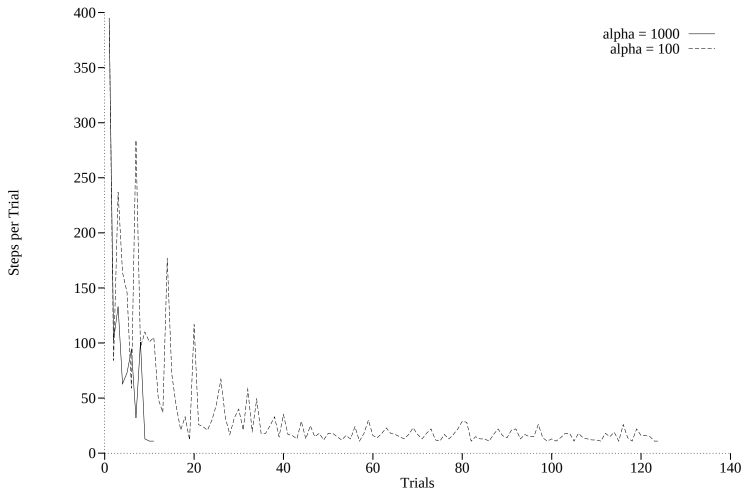


Figure 5.7: Learning Curve with 10 Hypothetical Steps and varying α

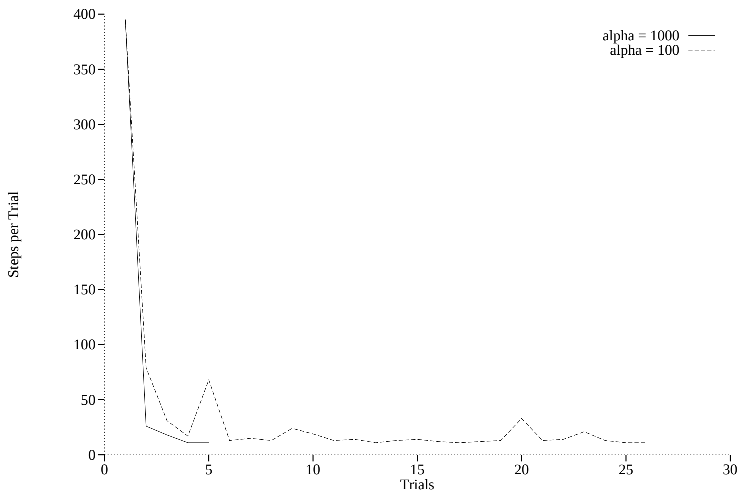


Figure 5.8: Learning Curve with 100 Hypothetical Steps and varying α

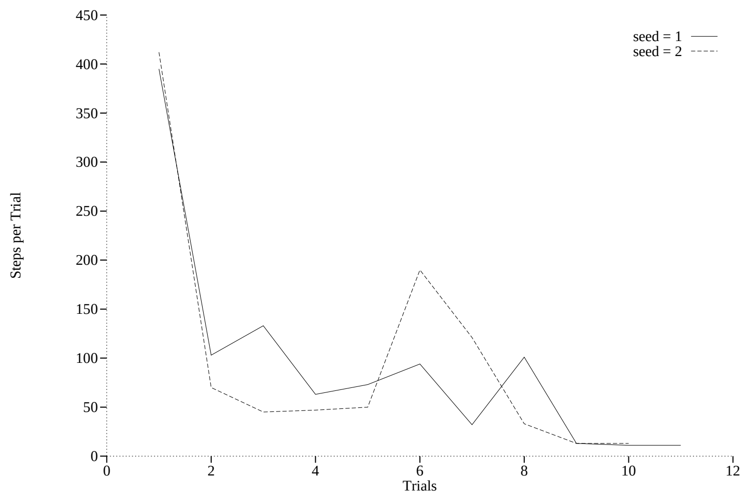


Figure 5.9: Learning Curve with 10 Hypothetical Steps, $\alpha = 1000$ and different seeds

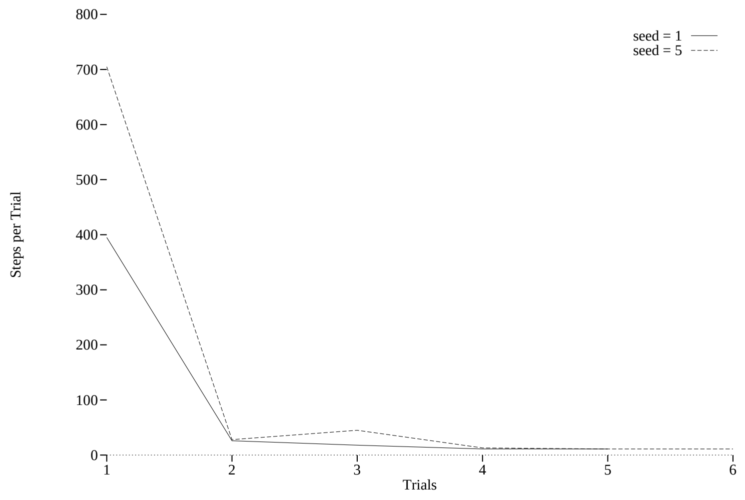


Figure 5.10: Learning Curve with 100 Hypothetical Steps, $\alpha = 1000$ and different seeds

of search problem typically studied in artificial intelligence. It does not show the full generality of the framework, in which rewards may be received on any step and there need not be a start or termination state. In the general case, the Dyna algorithm attempts to maximize the cumulative reward received per time step.

Despite these limitations, the results presented here are significant. They show that the use of an internal model can dramatically speed up trial-and-error learning processes even on simple problems and using a simple reward. Moreover, they show how the functionality of planning can be obtained in a completely incremental manner, and how a planning process can be freely intermixed with reactive and learning processes.

It shows that it is not necessary to choose between planning systems, reactive systems and learning systems. These three can be integrated not just into one system, but into a single algorithm, where each appears as a different facet or slightly different use of that algorithm.

Chapter 6

Experiments with the Real Robot

6.1 What is the difference ?

Experiments with the real robot requires attention to some additional points, that are ignorable during simulation. The real robot detects obstacles by sonar scanning, positioning is through odometry and needs to execute an evasive action when it finds an obstacle while moving from one position to another. The robot has an acceleration and deceleration phase in the movements and it cannot stop instantly. There are errors in the sensor readings that must be taken into account. The robot must return to its start position at the beginning of a new trial. Finally, the localization of the goal could be done using other kinds of sensors.

6.2 Additional Tasks

Sonar ranging, rotation and translation movements including the procedure for an evasive action (that means return the robot to the previous position in the case there was an obstacle in the way not detected before the beginning of the motion) were realized using some Albatros primitives.

To bring the robot back to its start position at the end of a trial and make eventual corrections in the case of non acceptable odometric errors a joystick was used to operate the robot manually.

6.2.1 Albatros Primitives

The Albatros operating system offers a set of standard commands which can be invoked from a remote machine, the operator or an application program. A command has a 4 Byte mnemonic followed by a list of parameters. The commands used here are:

MOTN

There are a set of commands that can be used to move the robot. The primitive chose here is the `MOTN` command. It allows to easily achieve the basic movements of the robot, i.e. translations and rotations around the central axis (center of driving axis), by specifying only a distance and a velocity. The command divides every move in elementary steps, taking care of acceleration and deceleration gradients. An obstacle detection process may be activated, but experiences showed that the lack of adjustable parameters made this approach of obstacle detection useless for this experiment.

Among the parameters ON, OFF, LEFT, RIGHT, FORWARD, BACKWARD there is the MOTN KILL that aborts the current movements with a deceleration phase. The MOTN STATUS that returns two values, one shows whether a motion is being executed and the other gives the distance traversed since the start of movement.

JSTK

With the parameters ON or OFF, activates a process that controls the use of the joystick to manually operate the platform.

RNOD

Specifies the mode of sonar sensor reading, which can be request or automatic as explained in Section 3.1.1. The mode used here is the request mode where each sensor can be read independently with a READ command.

READ

Is the read primitive of the ultrasonic device.

TSYS

Returns a pointer to some of the internal working tables of Albatros, used here to get the system time.

ERR

Makes a redirection of error messages from the standard output to any memory location. It is useful for debugging.

6.2.2 Sonar

The values returned from a sensor read correspond to the time elapsed between emission and reception of an ultrasonic wave. The distance can be calculated by multiplying the returned value with the period of the counter and then multiplying it with 300 m/s.

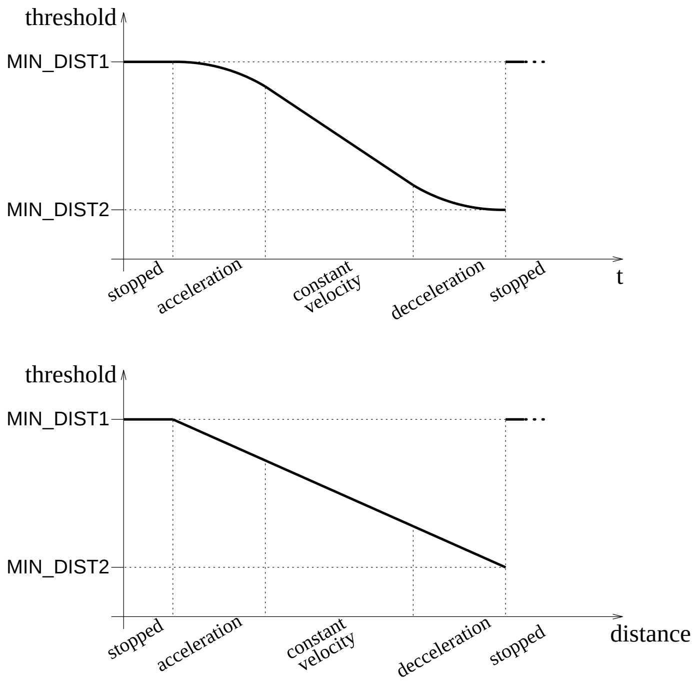


Figure 6.1: Sonar Scanning Threshold

The returned value however changes depending on the type of the objects, external temperature, etc.

Sonar Calibration

A calibration was made by positioning the objects at a certain distance reading sensor data and using that value as threshold.

Before the robot starts a movement it scans the direction it is going to move along using a higher threshold, proportional to the distance traversed. The threshold diminishes until a minimum value. See Figure 6.1, the values used here were approximately 1500mm and 20mm for `MIN_DIST1` and `MIN_DIST2` respectively.

Because the rotation axis of the robot is not located over the center of the platform but over the center of the driving axis, the threshold used cannot be the same for all directions relative to the robot. Thus, a margin (`DIST_OS`) is added when the directions to scan is not the front side as shown in Figure 6.2.

6.3 Module DynAlg

Although the module containing the algorithm remains the same, the software now runs on an Albatros machine and not on a workstation. That means the linking is

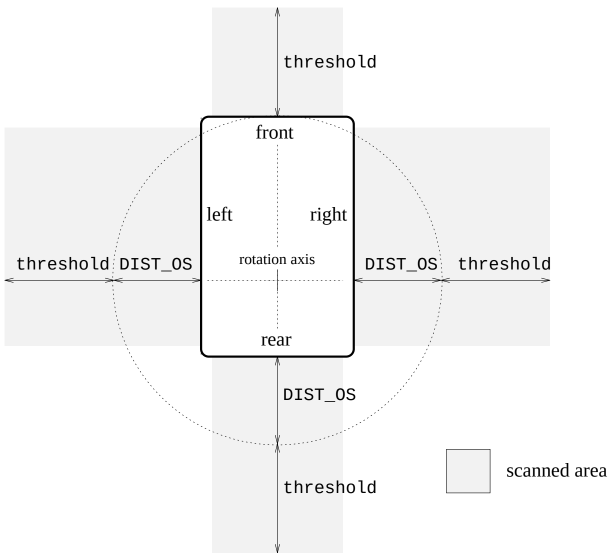


Figure 6.2: Corrections of Threshold

made using the Albatros libraries instead of the C standard libraries as described in Chapter 3.

Since there are no files in the Albatros machine to save data, the command *fprintf()* is not supported. Therefore it was necessary to make some changes in the module `DynAlg`. These changes concern only the saving and printout of the results, policy and evaluation tables and maps.

In order to adapt the module to run on the Albatros machine as discussed in Section 6.1, some changes had to be made in the files.

6.3.1 Defined Constants

The additional defined constants in `DynAlg.h` are:

TRIALMAX – maximum number that can be entered by the user for the maximum number of trials. This is necessary for the declaration of `Map`.

EVALU, POLICY – constants used in the `PrintResults` routine.

6.3.2 Global Variables

Additional global variables in `DynAlg.h` are:

Map – saves the `Frequency` of the cells of the world model for each trial. It contains all the information to draw the map. It is used by the routine `DrawMap`.

MoveCounter – counts the number of translations executed by the real robot in a trial, see Section 6.5.

DoActionsCounter – counts the number of times the `DoRealAction` function is invoked in a trial, see Section 6.5.

Simul – switcher variable that allows to run the algorithm in simulation mode on the Albatros machine. The robot is not moved and the obstacles are simulated. See function `DoRealAction`.

SimObst – switches on the simulation of obstacles defined in the function `GenerateSonarAlarm` even by using the robot (`Simul` to `OFF`.

6.3.3 Functions

The new or modified functions are:

DoRealAction(RealState,Action,NextState) – modified in order to allow the complete simulation of the task or only the simulation of some obstacles depending on the settings of `Simul` and `SimObst`.

MakeTables(Table) – modified in order to write the tables interactively on the standard output.

DrawMap(Trial) – saves the `Frequency` contained in `WorldModel` to `Map` at the end of each trial.

PrintResults(Trial) – prints the results in the to the standard output interactively, calls the functions `MakeTables` and `ShowMap` for this.

ShowMap(Trial) – now modified to use the standard output.

main() – modified to manage the differences in the other functions.

6.4 Module Movements

Once the module DynAlg was working correctly on the Albatros machine, the next step consisted in developing a new module *Movements* with all the new tasks necessary to move the robot, make a relative localization and perform obstacle detection. This module receives only the commands *Rotate()* and *Translate()* from the module *DynAlg*, that made it very easy to substitute the module *DynAlg* in the developing phase by a simpler main program that have only some rotation and translation commands, like in the file `passear.c` shown in Appendix C.

The module `Movements` consists of the file `Movements.h` and `Movements.c`. Listings are in Appendix C. It substitutes the preliminary module with the same name used in the simulation phase by a set of routines and functions that control a real robot and return the values depending on the circumstances of the real world.

6.4.1 Defined Constants

The defined constants in the header file are:

ROTAT_VEL – rotation velocity of the robot in deg/s.

TRANS_VEL – translation velocity of the robot in mm/s.

DISTANCE – length of one translation step in mm.

MIN_DIST1 – maximum threshold for the detection of an obstacle by the sonar cells, consists of the number of impulses of a counter.

MIN_DIST2 – minimum threshold for obstacle detection by sonar.

DIST_OS – value added to the threshold if scanning on the left, right or rear side.

MIN_DELAY – delay between sonar firings. The constant indicates the number of clock periods between two firings.

6.4.2 Global Variables

Some global variables are defined in the header file:

MotnStatus[2] – array to store the return values of the Albatros standard command MOTN STATUS, described in Section 6.2.1.

Error – address of the last Albatros error message. See Albatros standard command ERR.

DebugOn – boolean that activates a printout of status information for debugging.

SonarOn – boolean that switches the sonar scanning on or off.

Robot – struct variable that contains all the status information of the robot and the communication with Albatros. It works as the common area (common memory) between the main process and the motion process. The structure is arranged as follows:

- **Communic** – represents the communication status between the main process and `MotionProcess`. It can assume the values `CLEAR, NEW_COMMAND, ACK, EM_STOP` or `INITIALIZE`.
- **Code** – indicates the status about a motion command running on Albatros. The values can be `EXECUTING, READY` or `ABORTED`.
- **Status** – shows the status of the robot. Can be `INIT, OK, OBSTACLE` or `TROUBLE`.
- **MotionType** – can be `ROTATION` or `TRANSLATION`.
- **Direction** – associated to `LEFT, RIGHT, FORWARD` or `BACKWARD`.

Magnitude – contains the absolute distance or angular distance the robot will move.

Velocity – velocity of motion.

6.4.3 Functions in Movements.c

Functions used in `Movements.c`:

Rotate(Angle) – transforms the rotation angle into an angle less or equal to 180 degrees, performs a sonar scan in the direction the robot is supposed to turn and in order to perform the action, completes the variable `robot` and sends it as a packet to the `MotionProcess`.

Translate(Axis) – sends a packet to the `MotionProcess` and monitors `RobotStatus`. In the case of an obstacle sends a new packet to bring the robot back to the initial position. The axis parameter is used to correct an offset.

timer() – establishes a link to the Albatros system clock and returns the actual time.

MotionProcess() – interruptible process, see Section 6.4.4.

Scan(Side,Threshold) – makes a sonar scan in the direction of `Side` considering the time delay between firing set by `MIN_DELAY`, `Side` can be associated to `FRONTSD`, `BACKSD`, `RIGHTSD` or `LEFTSD`. Sets the sonar alarm on if an obstacle has been detected.

ReadFront(pDistance) – alternatively reads one of the three sensors of the front side of the robot each time the function is invoked. It writes the result in the array `pDistance`.

ReadBack(pDistance) – similar to `ReadFront`.

ReadRight(pDistance) – similar to `ReadFront`.

ReadLeft(pDistance) – similar to `ReadFront`.

InitMovements() – Initializes the `MOTN` command and starts the process `MotionProcess`.

ReturnRobot() – procedure to bring the robot back to the start position at the end of a trial. Turns `MOTN` off and activates the joystick.

TermMovements() – removes the `MotionProcess` and all other processes added to Albatros.

GenerateSonarAlarm() – simulates obstacles in the world. It is the same function used in the simulation phase.

PrintStatus() – prints the values of the `Robot` structure variables. The routine can be switched off through the `DebugOn` variable.

Report(TextLine) – writes the string `TextLine` to port 2, see Section 6.7.

GetText(TextLine) – reads string `Textline` from port 2, see Section 6.7.

6.4.4 The MotionProcess

The `MotionProcess` is a routine in the `Movements` module running as an independent and interruptible process. It is started every 10 clock cycles. It handles all the communication with Albatros concerning the translations and rotations, the emergency stop, controls the sonar scanning and continuously updates the status of the movement. The process is described in the flow chart on Figure 6.3 and 6.4.

In the chart the `RelPosError` means the error of the position of the robot relative to the desired end position.

The process runs parallel to the main process of the application program and communicates with it through a common area (global variable `Robot`). This has the advantage that the motion can be stopped at any time by another process or any external sensor that uses the same common area. The main process could execute another tasks while the robot is moving with the evasive action being controlled by the motion process.

There are basically three functions from the Albatros library *libcpr* used to add this interruptible process:

init_procit() initializes internal data structures and variables necessary for their management.

add_procit() adds a list of processes with attributes about priority, period and offset. The processes are listed in the data structure `Procs_To_Add`.

rem_procit() removes a list of processes contained in the data structure `Procs_To_Remove`.

The data structure `Procs_To_Add` and `Procs_To_Remove` are defined right before the main routine in the module `DynAlg`.

6.5 When does the robot execute the step

The number of real steps counted by the algorithm does not correspond to the number of movements executed by the robot. In Table 6.1 there is an experiment where the robot has to reach a goal that is only two steps away from the start position in a straight line. All other cells surrounding the path contain an obstacle. The first column shows the trial number, the second the number of real steps made by the algorithm in that trial, the third represents the number of times the function `DoRealAction` was called to execute the action and the last shows how many times the robot has really moved.

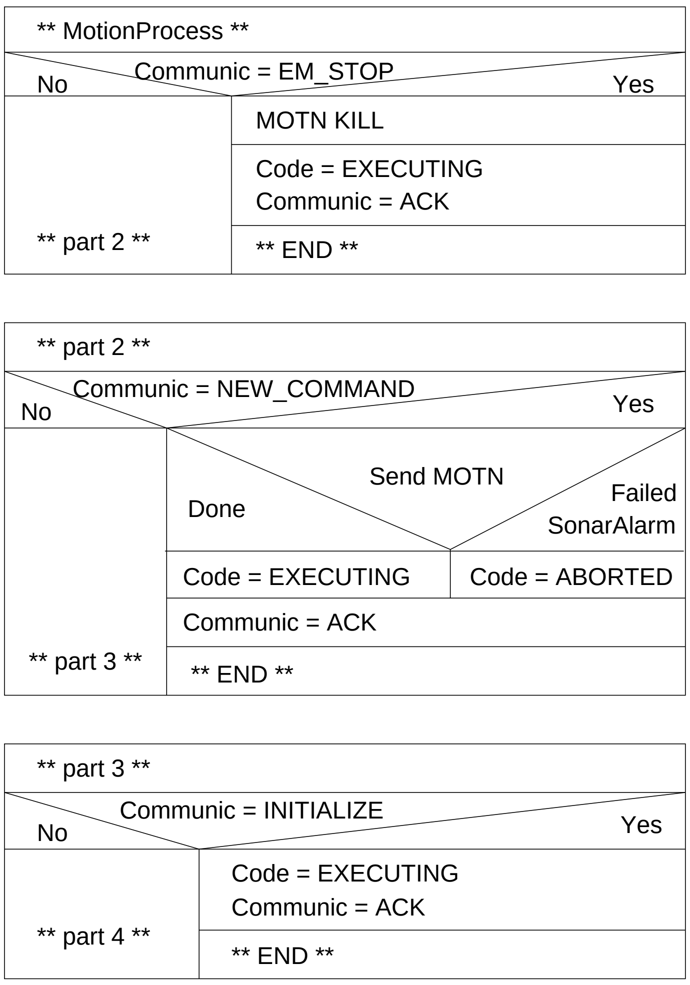


Figure 6.3: Flow Chart of MotionProcess

trial	steps	calls	motions
1	8	7	4
2	3	3	2
3	2	2	2
4	2	2	2

Table 6.1: Steps, Function Calls and Motions

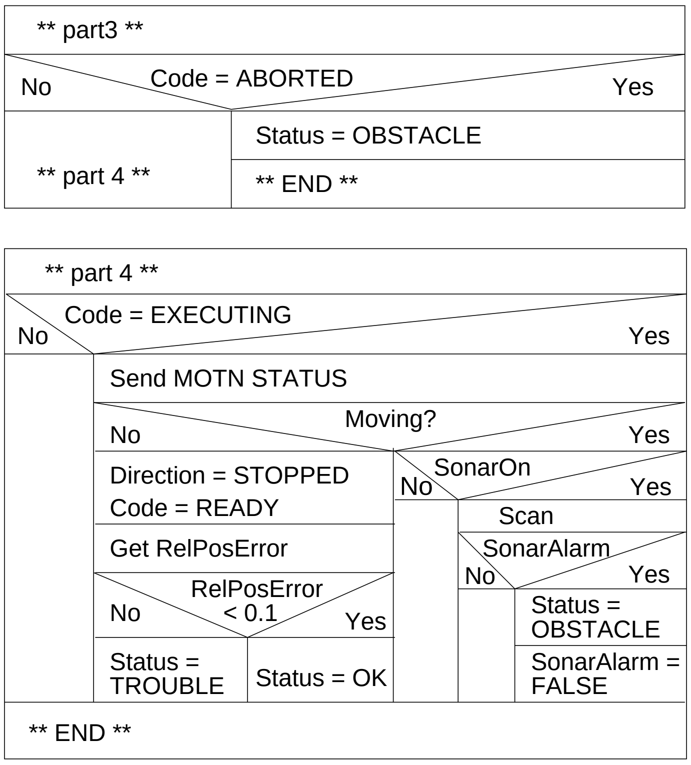


Figure 6.4: Flow Chart of MotionProcess (continuation)

The last two columns are different when the robot has detected an obstacle before starting the movement.

The action is only executed by the robot if there is no information of an obstacle, otherwise not even the sonars are switched on. If there is no information about the cell and the sonars detect an obstacle, the robot also doesn't move. In Figure 6.5 you can see where the counters are positioned to generate the result shown in Table 6.1. While the algorithm counts every real step, the function to perform a real step is only invoked if the new state is not OCCUPIED in the world model (see thick framed box `DoRealAction` in Figure 6.5), and the robot only translates if the sonars have not detected an obstacle before the rotation.

The number of motions with the robot should be as low as possible to avoid additional odometric errors. That is the reason for executing a sonar scan before moving.

6.6 Environment Used

The size of a cell is shown Figure 6.6. Although the length of the platform is only one meter the minimum size of a cell must be equal to 4 m^2. When the robot is positioned in a cell its rotation axis should be in the center of the square to give the robot space enough to turn around without colliding into a barrier. In addition, there is a safety margin of 20 cm. In Figure 6.7 an example of badly chosen cell sizes is shown. As you can see in the left example the robot cannot rotate without entering a neighbour cell, and in the right example the robot can have difficulties if it makes a 180 degree rotation to the right side. In this case the robot is located in a cell that normally is set as OCCUPIED and not allowed to be entered.

Obstacles are positioned in the world as shown in Figure 6.8. The walls of the room are used as obstacles. Wood fiberboards of the size of 120x50cm were positione vertically representing additional obstacles. Some obstacles are still simulated to avoid the robot running into obstacles that can not be detected by the sonars, e.g stairs, and to prevent the system from leaving the perimeter of the world model.

6.7 The Communication

Although the robot is autonomous and doesn't need to communicate with a remote machine, it is useful to get the data during the experiment to monitor the flow of the algorithm. This is especially true in the debugging phase.

To avoid using the serial cable to communicate, since this implies a physical link between the computer and the robot, a wireless radio modem was used for communication.

A wireless modem can be used instead of a cable: the modem is connected to a remote computer through a RS232 on the serial output. At the platform, a modem is installed in a similar way using one of the two ports of Albatros. This is shown in Figure 6.9.

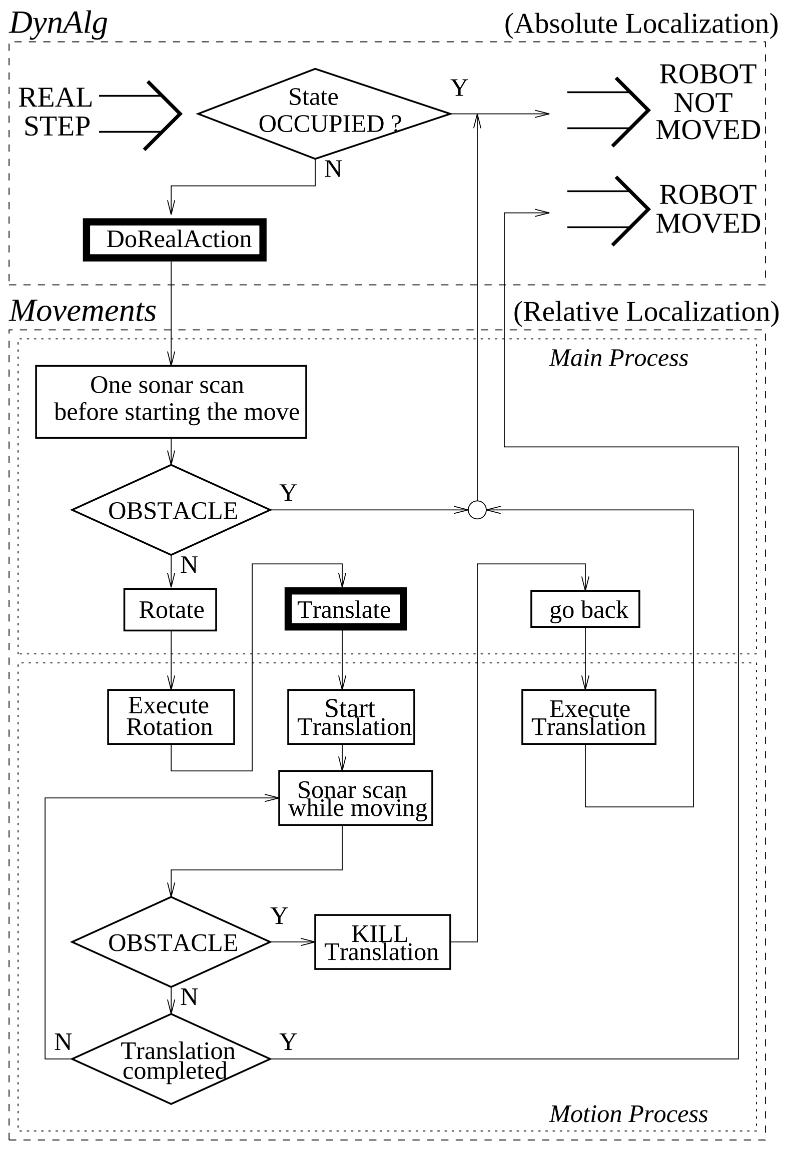


Figure 6.5: Motion Step

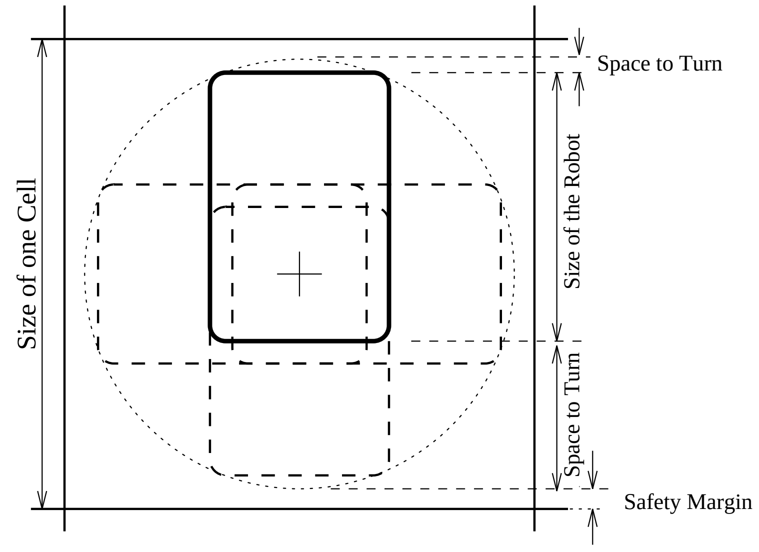


Figure 6.6: The Size of a Cell

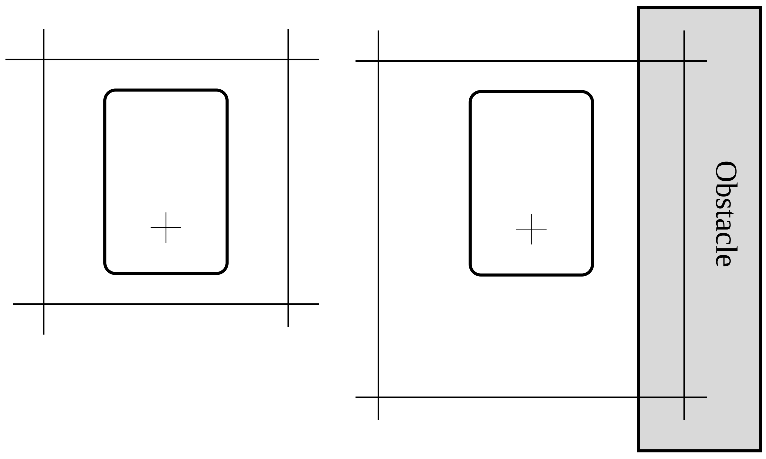


Figure 6.7: Badly Chosen Cells

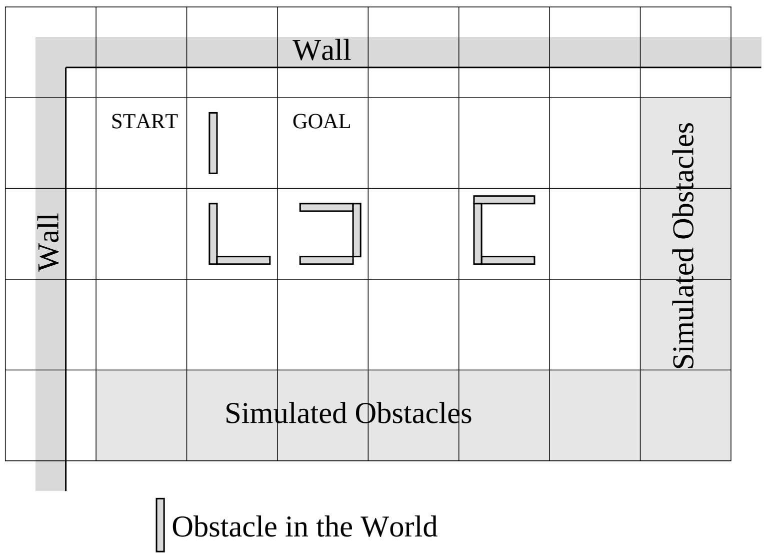


Figure 6.8: Obstacles in the World

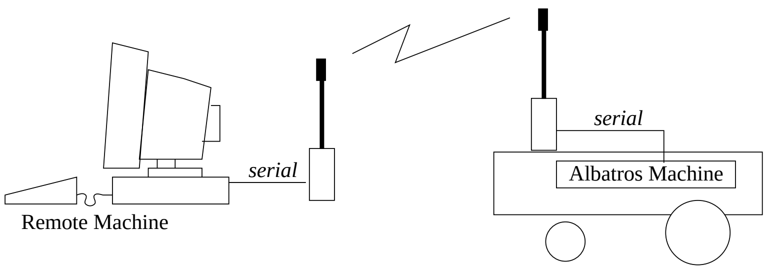


Figure 6.9: Using the Wireless Modem

The Albatros CPU has two serial ports that allow communication with a host computer. Since the transmittion of information through a serial line takes a long time (at 9600 baud, only one byte is transmitted every 1ms), a special software organization for asynchronous communication is provided in order to avoid any blocking of the whole application.

Serial line 1 drivers have been designed to manage a human operator dialog, while line 2 communicates with other computers. Port 2 has its own input buffer in which all received characters are stored. Serial line driver 1 always runs while port 2 must be validated, and every time a character arrives it is stored in an external buffer. It also has a mechanism to avoid overflow when the application software does not consume the messages fast enough.

Port 1 works as the output of all the standard input and output functions like *printf* and *scanf.* Apart from these standard functions there are a set of functions for the asynchronous communication. Those functions are related to:

- test functions to test the output and input status of the drivers.
- read and write functions to read or write a message from or to the driver buffer.
- open and close functions to validate or devalidate port 2.

Due to the hardware configuration of the robot, it was only possible to communicate via modem using port 2. An attempt to modify the hardware showed that it was more convenient to use port 2. Since it is not the standard communication port, the commands for asynchronous communication, that are more complicated to handle, were used. Unfortunately, modifications in the source code of the program were necessary to replace the the cable by the modem.

Since both the remote computer and the Albatros machine have two ports, cable and modem can be used simultaneously. The cable is used to download the program from the remote machine to Albatros and to start the program. Then the link through the cable is disconnected and the modem is used for monitoring the status.

6.8 Results

The complete results of the experiment can be seen in Appendix D.

The simulation of the algorithm in the Albatros machine provided exactly the same results as in the simulation by using the workstation, therefore there is no need to discuss it again.

One problem with the Albatros primitives has been detected. When starting a MOTN process after using the joystick and not resetting the system, the encoder shows a false result. The value provided by a MOTN STATUS command during the first MOTN process normally adds the distance run before with joystick, although in the second MOTN process the problem disappeared. The problem can be circumvented by making an ignorable small step of 1 mm after switching off the joystick.

The biggest problem have been the odometry errors. They may be kept small by reducing the movements of the robot. This is achieved by making sonar scans before the beginning of a movement.

After a MOTN command the distance driven by the robot is not exactly the same as the desired distance. The encoders always showed a slight difference around 10 mm less. To correct this, the difference was stored and added to the next movement in the same direction. This showed a good result in the translations but in the rotations was almost useless.

Chapter 7

Conclusion

The algorithm developed here enables a mobile robot to find the goal in an unknown environment and, in a sequence of trials, find the shortest path to reach it. It is based on iterative trial and error learning and planning steps using a self built world model and a policy that is continuously updated. All data is stored in the form of tables.

The simulation showed good functionality of the algorithm even with the restrictions imposed. The learning curve falls faster the more hypothetical steps are performed. It always converges without getting caught in local minima. The random number generator has a large influence in the duration of the experiment since it is used to choose the actions and to determine the states for the hypothetical steps.

The second part of this work was the experiment with the robot. It showed the complexity of adapting the task, first implemented in a simulated ideal environment, to work in a real environmetn where the conditions and circumstances are only approximated. It is now necessary to control the robot, perform the movements, evasive actions, sonar scanning and odometry, deal with inaccurate sensor data and accumulated errors. The results of the algorithm, however, were exactly the same as the ones obtained in the simulation. Regarding the fact that the robot needs not move on every real step, because some steps can be blocked in higher levels of the software by detecting an obstacle when consulting the world model or scanning an obstacle before moving the robot in it's direction.

The method of reward used makes the first trial very time consuming and monotonous since the system is not learning and planning but only finding the position of obstacles and goal. It learns only one policy step that is the last before entering the goal.

To speed up the first trial one could make use of other approaches, such as lateral scanning instead of scanning only in the direction of movement and store the encountered obstacles in the world model. Alternatively, a completely different algorithm for the first trial by punishing the actions that bring the robot back to the start could be used. We could even use the same algorithm and apply negative rewards for that purpose, causing the evaluation of a cell close to an obstacle and it's neighbouring cells to decrease. Consequently the robot is kept as far away as possible from obstacles. In most cases this is not the shortest but probably the safest way to reach the goal.

The policy which is not altered when the robot sees an obstacle could be updated by a different learning process without modifying the evaluation of a cell.

Nevertheless, the use of any other algorithm in the first trial to make the robot find the goal more effectively must be done carefully. In general, it requires a prior knowledge of the world concerning its size and empty spaces or the size and shape of the obstacles in order not to force the robot to perform actions with the opposite effect, bringing it away from the goal.

A test could be made by using bigger cells in the beginning of an experiment just to find the goal more easier and later divide every cell of the world into smaller ones. That, however, could diverge with the idea that once an obstacle has been located in a cell the entire cell is assumed to be occupied until the end of the algorithm. For that reason, the size of the cells must be chosen in such a way that there is no obstacle in the start cell and the goal, as it is shown in Figure 6.7, where the robot can see an obstacle in the cell and assume it to be occupied.

Large odometric failures can shift or twist the world model in such a way that it simply doesn't model the real world anymore. The robot then sees an obstacle in one cell which in the beginning was located in a neighbour cell. This situation can be dangerous if there is an obstacle close to the goal. The offset between the world model in which the robot always assumes to be in the center of a cell and the real position of the robot can differ so much that the robot could be very close to or even crossing the border to a neighbour cell. The result is that the robot can set the goal state as being occupied and never reach it again.

The use of parallel processing by performing the hypothetical steps simultaneously to a real step was not very interesting for this experiment specifically due to the robot used. Its large size and slow speed makes the time spent to execute the hypothetical steps neglectable compared to the duration of a real step in which the robot is moved. Parallel processing would contribute only to make the algorithm more elegant. However, when using a smaller and quicker robot such an approach would be interesting.

The use of beacons and a infrared camera could improve not only the localization but also permit the robot to self-recognize the goal.

Bibliography

[Sutt 90] Miller III, W.T.; Sutton, R. S. and Werbos, P. J.:
Neural Networks for Control.
MIT Press, 1990.

[Bart 83] Barto, A. G.; Sutton R. S. and Anderson C. W.:
Neuronlike elements that can solve difficult control problems.
IEEE Transcatiosn on Systems, Man and Cybernetics, SMC-13(5):835-846, 1983.

[Fu 70] Fu, K. S,:
Learning Control Systems - review and outlook.
IEEE Transactions on Automatic Control, AC-15(2), 1970.

[Fu 66] Fu, K. S. and Nikoliç, Z. J.:
On some reinforcement techniques and their relation to the stochastic approximation.
IEEE Transactions on Automation Control, AC-11(2):756-758, 1966.

[Lin 94] Lin, Long-Ji:
Scaling up reinforcement learning for robot control.
Proceedings of the Tenth International Conference on Machine Learning, 1994.

[Nar 89] Narendra, K. S. and Thathchar, M. A. L.:
Learning Automata - an Introduction.
Prentice Hall, 1989.

[Sin 92] Singh, S. P.:
Transfer of learning by composing solutions of elemental sequential tasks.
Machine Learning, 8:323-339, 1992.

[Tham 94] Tham, C. K. and Prager, R. W.:
A Modular Q-Learning Architecture for Manipulator Task Decomposition.
Proc. of the 11th International Conf. on Machine Learning.
Morgan Kaufmann Publischers Inc, 1994.

[Alb 89] Robosoft:
Albatros Reference Manual.

[Alb 91] Robosoft:
Albatros User's Manual.

[laun 89] Robosoft:
Local Area Ultrasonic Network, User's Manual.

[Pol] Polaroid:
Ultrasonic Ranging System, Description and Operation.

[Rob 91] Robosoft:
Robuter User's Manual.

Index

Appendix A

Source Code for Simulation

A.1 File README (Operating Instructions)

```
DESCRIPTION AND OPERATING INSTRUCTIONS FOR simulationV1.0

This directory contains the files necessary to simulate the
algorithm developed in the work:

Application of an Integrated Architecture for Learning,
Planning and Reacting on a Mobile Robot.

The files are:
- DynAlg.h
- DynAlg.c
- Movements.c
- Makefile
- do

Note: Lines beginning with '>' represent the lines that appear
on the screen of your computer when you run the program.
e.g.: >kim{weiseral}1:

Before starting:

1. make sure that the world model is in the size you want. You
   can set this in the DynAlg.h file by changing XMAX and YMAX.

2. you can also adjust the values of ALFA and BETA in DynAlg.h.

3. in the file Movements.c, routine GenerateSonarAlarm() you
   can introduce the obstacles, be sure to keep the borders of
   the world model occupied otherwise the robot could leave the
   area covered by the model.

   BUG: the algorithm does not check whether the robot has left
        the area covered by the world model.

4. make sure that the initial position assigned to the robot in
   the routine SetParameters() of DynAlg.c is correct, inclu-
   sive the angle. Even in the simulation different initial
   angles lead to different results. You could modify the rou-
   tine SetParameters() in order to set the initial position
   interactively.
```

Compile the code and start the program:

The script 'do' compiles and links the code by invoking the Makefile and starts the program. At the end it edits the results of the experiment.
To start the script 'do' type:

```
>kim{weiseral}21: do
```

You will see:

```
>Initializing...
```

Then you have to enter the number of hypothetical steps to be executed after each real step:

```
>Enter number of hypothetical cycles: 100
```

Enter the maximum number of trials, if the algorithm hasn't found the optimal path until that trial it will be stopped.

```
>Enter number of trials: 20
```

Enter a seed number to initialize the random number generator.

```
>Enter seed (unsigned short): 1
```

The start state you have already if not modified to make it interactively.

```
>Enter start state (x, y, ang): (2, 5, 270)
```

Enter the goal state:

```
>Assign goal state (x,y): 8 4
>(8,4)
```

The used values for alpha and beta are printed here:

```
>ALFA = 1000
>BETA = 100/1000
```

Now the algorithm is running:

```
>Working...
```

If the optimal path has been found before the algorithm reached the maximum number of trials you will see something like this.

```
>Optimal path found in 4 trials.
>Need 10 steps to reach goal.
```

If no optimal path was found.

```
>Couldn't find optimal path.
```

Since the program ended the script 'do' opens three windows showing the results and the prompt appears again.

```
>[1] 13249
>[2] 13250
```

```
>[3] 13251
>kim{weiseral}22:

One window shows the file PathMap, the frequency the robot has
been in each state. The sign # represents an obstacle. Here you
can see the position of the obstacles.
Some clippings of an example:

>
> REAL EXPERIENCE 1 - 394 steps
>
>
>
>          #   #
> #   8  23  14   #   #       #   1
> #  19  19  18   #   8   1   4   1
> #  31  18   #   #   5   3   3
> #  21  25  13  12   8   7
> #  18  33  27  22  18  10   5
>     #   #   #   #   #       #
>

> REAL EXPERIENCE 5 - 10 steps
>
>
>
>          #   #
> #               #   #       #   1
> #               #   1   1   1   1
> #       1   #   #   1
> #       1   1   1   1
> #
>     #   #   #   #   #       #
>

Other window shows the file RobotTables containing the updated
evaluation and policy tables.

>Evaluation Table
>
> 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
>
> 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
>
> 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
>
> 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
>
> 0.000 0.257 0.293 0.243 0.000 0.000 0.000 0.000 0.000 0.000
>
> 0.000 0.303 0.340 0.269 0.000 0.729 0.810 0.900 1.000 0.000
>
> 0.000 0.342 0.384 0.000 0.000 0.656 0.729 0.810 0.000 0.000
>
> 0.000 0.378 0.429 0.478 0.531 0.590 0.656 0.000 0.000 0.000
>
> 0.000 0.345 0.385 0.429 0.477 0.531 0.590 0.531 0.000 0.000
>
> 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
>
>
>Policy Table
```

```
>
>+ - - + - - + - - + - - + - - + - - + - - + - - + - - + - - +
>|  25 |  25 |  25 |  25 |  25 |  25 |  25 |  25 |  25 |  25 |
> 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
>|  25 |  25 |  25 |  25 |  25 |  25 |  25 |  25 |  25 |  25 |
>+ - - + - - + - - + - - + - - + - - + - - + - - + - - + - - +
>|  25 |  25 |  25 |  25 |  25 |  25 |  25 |  25 |  25 |  25 |
> 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
>|  25 |  25 |  25 |  25 |  25 |  25 |  25 |  25 |  25 |  25 |
>+ - - + - - + - - + - - + - - + - - + - - + - - + - - + - - +
>|  25 |  25 |  25 |  25 |  25 |  25 |  25 |  25 |  25 |  25 |
> 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
>|  25 |  25 |  25 |  25 |  25 |  25 |  25 |  25 |  25 |  25 |
>+ - - + - - + - - + - - + - - + - - + - - + - - + - - + - - +
>|  25 |  25 |  25 |  25 |  25 |  25 |  25 |  25 |  25 |  25 |
> 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
>|  25 |  25 |  25 |  25 |  25 |  25 |  25 |  25 |  25 |  25 |
>+ - - + - - + - - + - - + - - + - - + - - + - - + - - + - - +
>|  25 |   9 |   2 |  10 |  25 |  25 |  25 |  25 |  25 |  25 |
> 25 25  9 14  2  1 65 10 25 25 25 25 25 25 25 25 25 25 25 25
>|  25 |  69 |  95 |  16 |  25 |  25 |  25 |  25 |  25 |  25 |
>+ - - + - - + - - + - - + - - + - - + - - + - - + - - + - - +
>|  25 |   3 |   1 |   2 |  25 |   0 |   0 |   0 | 100 |  25 |
> 25 25  4  5  1  1 92  3 25 25  0100  0100  0100  0  0 25 25
>|  25 |  87 |  97 |   3 |  25 |   0 |   0 |   0 |   0 |  25 |
>+ - - + - - + - - + - - + - - + - - + - - + - - + - - + - - +
>|  25 |   2 |   1 |  25 |  25 |  99 |  92 | 100 |  25 |  25 |
> 25 25  4 76  1  1 25 25 25 25  0  0  0  8  0  0 25 25 25 25
>|  25 |  18 |  98 |  25 |  25 |   0 |   0 |   0 |  25 |  25 |
>+ - - + - - + - - + - - + - - + - - + - - + - - + - - + - - +
>|  25 |   1 |   0 |   0 |   0 |  99 | 100 |  25 |  25 |  25 |
> 25 25  1 97  0 99  0 99  0 99  0  0  0  0 25 25 25 25 25 25
>|  25 |   1 |   0 |   0 |   0 |   0 |   0 |  25 |  25 |  25 |
>+ - - + - - + - - + - - + - - + - - + - - + - - + - - + - - +
>|  25 |   2 |   2 |   8 |   1 |   6 | 100 |   0 |  25 |  25 |
> 25 25  2 93  1 95  1 88  1 98  1 92  0  0 99  0 25 25 25 25
>|  25 |   2 |   2 |   2 |   1 |   1 |   0 |   0 |  25 |  25 |
>+ - - + - - + - - + - - + - - + - - + - - + - - + - - + - - +
>|  25 |  25 |  25 |  25 |  25 |  25 |  25 |  25 |  25 |  25 |
> 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
>|  25 |  25 |  25 |  25 |  25 |  25 |  25 |  25 |  25 |  25 |
>+ - - + - - + - - + - - + - - + - - + - - + - - + - - + - - +

Other window shows the file RobotPath where it is possible to
see all the steps, real and hypothetical, done by the algorithm
Observe that in the first trial the algorithm makes only real
steps. The file shows in each line the
x-position, y-position and angle (only real steps)
Some clippings of an example:

>   TRIAL 1
>  2   4  90      Real Step    1
>  2   5 270      Real Step    2
>  3   5   0      Real Step    3
>  3   5 270      Real Step    4
>  .   .   .        .    .     .
>  .   .   .        .    .     .
>  .   .   .        .    .     .
>  7   5  90      Real Step  390
>  7   5  90      Real Step  391
>  7   6 270      Real Step  392
>  7   5  90      Real Step  393
>  8   5   0      Real Step  394
>  8   4  90      Real Step  395
>   394 iterations
>
>   TRIAL 2
>  2   4  90      Real Step    1
>  5   6          Hypo Step    1
>  3   5          Hypo Step    2
>  5   7          Hypo Step    3
>  .   .   .        .    .     .
>  .   .   .        .    .     .
>  .   .   .        .    .     .
>  4   3 270      Real Step   26
>  7   5          Hypo Step    1
>  .   .   .        .    .     .
>  .   .   .        .    .     .
>  .   .   .        .    .     .
>  5   5          Hypo Step   98
>  3   4          Hypo Step   99
>  4   8          Hypo Step  100
>    25 iterations
>
>   TRIAL 5
>  .   .   .        .    .     .
>  .   .   .        .    .     .
>  .   .   .        .    .     .
>  7   8          Hypo Step   93
>  8   5          Hypo Step   94
>  8   5          Hypo Step   95
>  1   6          Hypo Step   96
>  3   4          Hypo Step   97
>  7   6          Hypo Step   98
>  7   6          Hypo Step   99
>  6   8          Hypo Step  100
>    10 iterations
>
>
>Optimal path found in 4 trials.
>Need 10 steps to reach goal.
```

A.2 Makefile

```
#---------------------------------------------------------------
# Makefile for passear and later for DynAlg
#---------------------------------------------------------------
# Unit:
# File:
# Date:
# Author:
# Version:
# Revisions:
# Comments:
#
# Functions:
#---------------------------------------------------------------

dyna: DynAlg.o
cc DynAlg.o -lm -g -o DynAlg

DynAlg.o: DynAlg.h DynAlg.c Movements.c
cc -c  $<
```

A.3 File do

```
make dyna
DynAlg
#foreach parameter ($*)
#  if  ("$parameter" == "p") then
    xedit -g 250x800 RobotPath &
#  endif
#  if  ("$parameter" == "t") then
   xedit -g 950x600 RobotTables &
#  endif
#  if  ("$parameter" == "m") then
    xedit -g 600x700 PathMap &
#  endif
#end
```

A.4 File DynAlg.h

```
/*.............................................................
DynAlg.h - header file containing the global variables for
DynAlg.c, an Integrated Architecture for Learning, Planning and
Reacting.
...............................................................
Unit:
File:
Date:
Author:
Version:
Revisions:
Comments:

.............................................................*/
#define XMAX 10  /* number of WorldModelCells in x direction */
#define YMAX 10  /* number of WorldModelCells in y direction */
#define TRIALMAX 400    /* maximum number of trials          */
#define DX 1            /* magnitude of step in x directiono */
#define DY 1            /* magnitude of step in y directiono */
#define ACTIONS 4       /* number of different actions that can
                                 be taken from a state (cell)*/
#define STATEDIM 3      /* dimension of the state vector     */
#define GAMMA 900  /* /1000 used to calculate posterior eval */
#define ALFA 1000  /* used to update the policy table        */
#define BETA 100   /* /1000 used to update the evaluatn table*/

typedef int *pstate;    /* pointer to a state in world model */
typedef int state;      /* states in the world model         */
enum statecomp { X, Y, ANG };

typedef int boolean;    /* define boolean type               */
enum booleanset { FALSE, TRUE };

typedef int switcher;   /* define switcher type              */
enum switcherset { OFF, ON };

typedef int action;     /* action to be taken from a state   */
enum actionset { UP, DOWN, LEFT, RIGHT };

typedef int infotype;   /* informations for a cell in model  */
enum infoset { UNKNOWN, EMPTY, OCCUPIED };

struct worldcell {
  infotype Info; /*contains info about the cell as in infoset*/
  boolean ActionKnown[ACTIONS];
  int Reward;                /* reward for reaching the cell */
  int Frequence; /* contains the number of times the cell ...*/
} WorldModelCell[XMAX][YMAX];        /* ... has been reached */

int Map[TRIALMAX][XMAX][YMAX]; /* saves the worldmodel at...
                                           ... each  trial */

double Eval[XMAX][YMAX];       /* the evaluation table       */

double Policy[XMAX][YMAX][ACTIONS];  /* the policy table     */

state RobotPos[STATEDIM];  /* position of the robot for sonar*/
boolean SonarAlarm=FALSE;  /* obstacle avoidance             */
boolean GoalFound=FALSE;   /* set by infrared sensor         */
boolean GoalPointSet=FALSE;  /*goal has not been located yet */
int XGoal, YGoal;            /* position of the goal state   */
```

A.5 File DynAlg.c

```
/*..............................................................
  DynAlg.c - file containing the algorithm for Dyna,
             an Integrated Architecture for Learning, Planning
             and Reacting.
...............................................................
Unit:
File:  DynAlg.c
Date:  16 Jan 1995
Author:  Alex Weiser
Version:  0.1
Revisions:
Comments: used only for simulation until now.
  Creates the files RobotPath, RobotTables and PathMap.
  RobotPath contains the path followed during the learning
  process. RobotTables contains the evaluation, policy and
  world model tables. PathMap shows how the robot  builds the
  world model and the frequence the robot has been in each cell

Functions:
  RealStep(RealState)
  HypotStep(HypotState)
  Evaluate(State)
  ChooseAction(CurrState,pAction)
  GetNextState(State,Action,NextState)
  DoRealAction(RealState,Action,NextState)
  FormPostEval(NewState,ActionReward)
  UpdateWorldModel(State,NextState,Info,ActionReward)
  UpdatePolicyEval(State,PriorEval,PostEval,Action)
  GetReward(State)
  IsGoal(State)
  Initialise()
  SetParameters(pHypotCycles,pTrialMax,StartState)
  MakeTables()
  DrawMap(fp,Trial,Steps)
.............................................................*/

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "DynAlg.h"
#include "Movements.c"

/*..............................................................
RealStep(RealState)- routine to execute a step on the real
world.
.............................................................*/
void RealStep(RealState)
pstate RealState;
{
int i, ActionReward;
double PriorEval,PostEval;
action Action;
infotype StateInfo;
state NextState[STATEDIM], OldRealState[STATEDIM];
infotype GetNextState();
double Evaluate(), FormPostEval();
boolean DoRealAction(), ChooseAction();
void UpdateWorldModel(), UpdatePolicyEval();
extern struct worldcell WorldModelCell[XMAX][YMAX];
```

```
  PriorEval = Evaluate(RealState);

  ChooseAction(RealState,&Action);

  for(i=0;i<STATEDIM;i++) OldRealState[i] = RealState[i];

  StateInfo = GetNextState(RealState,Action,NextState);

  if (StateInfo!=OCCUPIED && DoRealAction(RealState,Action,
    NextState)){
    StateInfo = EMPTY;
    ActionReward = GetReward(RealState);
    PostEval = FormPostEval(NextState,ActionReward);
  }
  else{
    StateInfo = OCCUPIED;
    ActionReward = 0;
    PostEval = PriorEval;
  }

  UpdateWorldModel(OldRealState,NextState,StateInfo,
    ActionReward);
  UpdatePolicyEval(OldRealState,PriorEval,PostEval,Action);

} /* RealStep */

/*..............................................................
HypotStep(HypotState)- routine to execute a hypothetical step
on the world model.
............................................................*/
void HypotStep(HypotState)
pstate HypotState;
{
int ActionReward;
double PriorEval, PostEval;
action Action;
state NextState[STATEDIM];
infotype GetNextState();
boolean ChooseAction();
double Evaluate(), FormPostEval();
void UpdatePolicyEval();

  PriorEval = Evaluate(HypotState);

  do{
    ChooseAction(HypotState,&Action);
  }while(GetNextState(HypotState,Action,NextState) != EMPTY);

  ActionReward = WorldModelCell[NextState[X]][NextState[Y]].
    Reward;
  PostEval = FormPostEval(NextState,ActionReward);

  UpdatePolicyEval(HypotState,PriorEval,PostEval,Action);

} /* HypotStep */

/*..............................................................
Evaluate(State)- returns the heuristic reward written in the
evaluation table for the specified state.
............................................................*/
double Evaluate(State)
```

```
pstate State;
{
extern double Eval[XMAX][YMAX];

  return(Eval[State[X]][State[Y]]);

} /* Evaluate */

/*..............................................................
ChooseAction(State,pAction)- decides the action to be taken
from the given state by checking the policy table. In the case
that the policy table is empty for that state, the action is
chose randomly.
..............................................................*/
boolean ChooseAction(State,pAction)
pstate State;
action *pAction;
{
int i;
double RandNumb, ProbabSum, drand48();
double UpToDown, DownToLeft, LeftToRight;
extern double Policy[XMAX][YMAX][ACTIONS];
extern struct worldcell WorldModelCell[XMAX][YMAX];

  ProbabSum = 0.0;

  for(i=0;i<ACTIONS;i++)
    ProbabSum += exp(Policy[State[X]][State[Y]][i]);

  UpToDown = exp(Policy[State[X]][State[Y]][UP])/ProbabSum;
  DownToLeft = exp(Policy[State[X]][State[Y]][DOWN])/ProbabSum
    + UpToDown;
  LeftToRight = exp(Policy[State[X]][State[Y]][LEFT])/ProbabSum
    + DownToLeft;

  while(TRUE){
    RandNumb = drand48();
    if(RandNumb>0 && RandNumb<UpToDown)
      *pAction=UP;
    else if (RandNumb>UpToDown && RandNumb<DownToLeft)
      *pAction=DOWN;
    else if (RandNumb>DownToLeft && RandNumb<LeftToRight)
      *pAction=LEFT;
    else if (RandNumb>LeftToRight && RandNumb<1)
      *pAction=RIGHT;
    else
      continue;
    break;
  }

} /* Evaluate */

/*..............................................................
GetNextState(State,Action,NewState)- plans the following step
calculating the new position. Returns an integer value
according to the information about the new State.
..............................................................*/
infotype GetNextState(State,Action,NewState)
action Action;
pstate NewState, State;
{
```

```
int pos[STATEDIM];
extern struct worldcell WorldModelCell[XMAX][YMAX];

  switch(Action){
    case UP:
      pos[X] = State[X];
      pos[Y] = State[Y] - DY;
      break;
    case DOWN:
      pos[X] = State[X];
      pos[Y] = State[Y] + DY;
      break;
    case LEFT:
      pos[X] = State[X] - DX;
      pos[Y] = State[Y];
      break;
    case RIGHT:
      pos[X] = State[X] + DX;
      pos[Y] = State[Y];
      break;
  }

  NewState[X] = pos[X];
  NewState[Y] = pos[Y];
  NewState[ANG] = State[ANG];

  return(WorldModelCell[NewState[X]][NewState[Y]].Info);

} /* GetNextState */


/*.........................................................
DoRealAction(RealState,Action,NextState)- executes the real
action with the robot and returns a boolean value according
to the success of the action, i.e., presence or not of an
obstacle.
.......................................................*/
boolean DoRealAction(RealState,Action,NextState)
action Action;
pstate RealState, NextState;
{
int Angle, AngleDiff;
void Rotate();
boolean Transalte();
void GenerateSonarAlarm();

  switch(Action){
    case UP:
      Angle = 90;  break;
    case DOWN:
      Angle = 270; break;
    case LEFT:
      Angle = 180; break;
    case RIGHT:
      Angle = 0;   break;
  }

  AngleDiff = Angle-RealState[ANG];

  GenerateSonarAlarm(RealState,Action);
  Rotate(AngleDiff);
  RealState[ANG] = Angle;

  if(Translate()){
    RealState[X] = NextState[X];
    RealState[Y] = NextState[Y];
    return(TRUE);
  }
  else
    return(FALSE);

} /* DoRealAction */

/*..............................................................
FormPostEval(NewState,Reward)- recalculates the evaluation of
the old state. Value used fo eventual corrections in the
policy.
..............................................................*/
double FormPostEval(NewState,Reward)
pstate NewState;
int Reward;
{
double PostEval;
double Evaluate();

  PostEval = (double)Reward + (double)GAMMA/1000.0 *
    Evaluate(NewState);

  return(PostEval);

} /* FormPostEval */

/*..............................................................
UpdateWorldModel(State,NextState,Info,ActionReward)- writes
the reward, learned from the real world, in the world model.
..............................................................*/
void UpdateWorldModel(State,NextState,Info,ActionReward)
pstate State,NextState;
infotype Info;
int ActionReward;
{
extern struct worldcell WorldModelCell[XMAX][YMAX];

  WorldModelCell[NextState[X]][NextState[Y]].Info = Info;
  WorldModelCell[NextState[X]][NextState[Y]].Reward =
    ActionReward;

  if (Info==EMPTY)
    WorldModelCell[NextState[X]][NextState[Y]].Frequence++;
  else
    WorldModelCell[State[X]][State[Y]].Frequence++;

} /* UpdateWorldModel */

/*..............................................................
UpdatePolicyEval(State,PriorEval,PostEval,Action)- updates the
policy and the evaluation table.
..............................................................*/
void UpdatePolicyEval(State,PriorEval,PostEval,Action)
pstate State;
double PriorEval, PostEval;
action Action;
```

```
{
double EvalDiff, PolicyIncr, Decrement;
extern double Eval[XMAX][YMAX];
extern double Policy[XMAX][YMAX][ACTIONS];

  EvalDiff = PostEval - PriorEval;
  Eval[State[X]][State[Y]] += (double)BETA/1000.0 * EvalDiff;
  PolicyIncr= (double)ALFA/1000.0 * EvalDiff;
  Policy[State[X]][State[Y]][Action] += PolicyIncr;

} /* UpdatePolicyEval */

/*..............................................................
GetReward(State)- returns the reward of the new state.
The reward should be supplied by a reaction from the real world
.............................................................*/
int GetReward(State)
state State[STATEDIM];
{
extern boolean GoalPointSet;

  if (GoalPointSet && State[X]==XGoal && State[Y]==YGoal)
    return(1);
  else
    return(0);

} /* GetReward */

/*..............................................................
IsGoal(State)- returns TRUE when the state corresponds to the
goal.  First it checks the signal 'GoalFound' and then the last
values in XGoal and YGoal.
.............................................................*/
boolean IsGoal(State)
pstate State;
{
extern int XGoal, YGoal;
extern boolean GoalFound, GoalPointSet;

  if (GoalFound){    /* boolean value set by infrared sensor */
    XGoal = State[X];
    YGoal = State[Y];
    GoalFound = FALSE;
    GoalPointSet = TRUE;
    return(TRUE);
  }
  else if (GoalPointSet && State[X]==XGoal && State[Y]==YGoal)
    return(TRUE);
  else
    return(FALSE);

} /* IsGoal */

/*..............................................................
Initialise()- initialises the evaluation table, the policy
table and the world model.
.............................................................*/
void Initialise()
{
int i, j, k;
extern double Eval[XMAX][YMAX];
extern double Policy[XMAX][YMAX][ACTIONS];
extern struct worldcell WorldModelCell[XMAX][YMAX];
void InitMovements();

  printf("\nInitialising...\n");

  for(i=0;i<XMAX;i++){
    for(j=0;j<YMAX;j++){
      Eval[i][j]=0.0;
      WorldModelCell[i][j].Info=UNKNOWN;
      WorldModelCell[i][j].Reward=0;
      WorldModelCell[i][j].Frequence=0;
      for(k=0;k<ACTIONS;k++){
        WorldModelCell[i][j].ActionKnown[k]=FALSE;
        Policy[i][j][k]=0.0;
      }
    }
  }

  InitMovements();

} /* Initialise */

/*...............................................................
SetParameters(pHypotCycles,pTrialMax,StartState)- this routine
prompts the user to enter parameters.
.............................................................*/
void SetParameters(pHypotCycles,pTrialMax,StartState)
int *pHypotCycles, *pTrialMax;
pstate StartState;
{
int Seed;
extern int XGoal, YGoal;
void srand48();

  printf("\nEnter number of hypothetical cycles: ");
  scanf("%d",pHypotCycles);

  do{
    printf("\nEnter number of trials: ");
    scanf("%d",pTrialMax);
  }while(*pTrialMax>TRIALMAX);

  printf("\nEnter seed (unsigned short): ");
  scanf("%d",&Seed);
  srand48(Seed);

  printf("\nEnter start state (x, y, ang): ");
  StartState[X]=2;
  StartState[Y]=5;
  StartState[ANG]=270;
  WorldModelCell[StartState[X]][StartState[Y]].Info = EMPTY;

  printf("(%d, %d, %d)\n",StartState[X],StartState[Y],
    StartState[ANG]);

  printf("\nAssign goal state (x,y): ");
  scanf("%d  %d",&XGoal,&YGoal);
  GoalPointSet = TRUE;
  printf("(%d,%d)\n",XGoal,YGoal);

} /* SetParameters */
```

```
/*.............................................................
MakeTables()- writes the values of the evaluation table, the
policy table and the world model in the file RobotTables.
.............................................................*/
void MakeTables()
{
int i,j, k, PolicyXBegin;
double ProbabSum[XMAX];
FILE *fpTables;
extern double Eval[XMAX][YMAX];
extern double Policy[XMAX][YMAX][ACTIONS];
extern struct worldcell WorldModelCell[XMAX][YMAX];

  fpTables = fopen("RobotTables", "w");

  /* Evaluation Table */
  fprintf(fpTables,"Evaluation Table\n\n");
  for(k=0;k<YMAX;k++){
    for(i=0;i<XMAX;i++){
      fprintf(fpTables," %0.3f",Eval[i][k]);
    }
    fprintf(fpTables,"\n\n");
  }

  /* Policy Table */
  PolicyXBegin=0;
  fprintf(fpTables,"\nPolicy Table\n\n",PolicyXBegin);
  /*fprintf(fpTables,"      up\n left   right\n     down\n");*/

  for(i=PolicyXBegin;i<XMAX;i++){
    fprintf(fpTables,"+ - - ");
  }
  fprintf(fpTables,"+\n");

  for(k=0;k<YMAX;k++){
    for(i=PolicyXBegin;i<XMAX;i++){
      ProbabSum[i] = 0.0;
      for(j=0;j<ACTIONS;j++)
        ProbabSum[i] += exp(Policy[i][k][j]);
    }

    for(i=PolicyXBegin;i<XMAX;i++){
      fprintf(fpTables,"| %3.0f ",
        exp(Policy[i][k][UP])/ProbabSum[i]*100.);
      /*fprintf(fpTables,"| %3.0f  ",
        exp(Policy[i][k][UP])/ProbabSum[i]*100.);*/
      /*fprintf(fpTables,"|  %5.1f   ",
        exp(Policy[i][k][UP])/ProbabSum[i]*100.);*/
    }
    fprintf(fpTables,"|\n");

    for(i=PolicyXBegin;i<XMAX;i++){
      fprintf(fpTables,"%3.0f%3.0f",
        exp(Policy[i][k][LEFT])/ProbabSum[i]*100.,
        exp(Policy[i][k][RIGHT])/ProbabSum[i]*100);
      /*fprintf(fpTables," %5.1f%5.1f",
        exp(Policy[i][k][LEFT])/ProbabSum[i]*100.,
        exp(Policy[i][k][RIGHT])/ProbabSum[i]*100);*/
    }
    fprintf(fpTables,"\n");

    for(i=PolicyXBegin;i<XMAX;i++){
      fprintf(fpTables,"| %3.0f ",
        exp(Policy[i][k][DOWN])/ProbabSum[i]*100);
      /*fprintf(fpTables,"|  %5.1f   ",
        exp(Policy[i][k][DOWN])/ProbabSum[i]*100);*/
    }
    fprintf(fpTables,"|\n");

    for(i=PolicyXBegin;i<XMAX;i++){
      fprintf(fpTables,"+ - - ");
    }
    fprintf(fpTables,"+\n");
  }
  fprintf(fpTables,"\n");

  fclose(fpTables);

} /* MakeTables */


/*..............................................................
DrawMap(fp,Trial,Steps)- draws a map with the path followed by
the robot in the last real experience.
..............................................................*/
void DrawMap(fp,Trial,Steps)
FILE *fp;
int Trial, Steps;
{
int i, k;
  /*fprintf(fp,"\n%d\t\t%d ",Trial,Steps);*/

  fprintf(fp,"\n REAL EXPERIENCE %d - %d steps\n",Trial,Steps);

  for(k=0;k<YMAX;k++){
    for(i=0;i<XMAX;i++){
      if(WorldModelCell[i][k].Info==EMPTY){
        if(WorldModelCell[i][k].Frequence!=0)
          fprintf(fp,"%2d  ",WorldModelCell[i][k].Frequence);
        else
          fprintf(fp,"    ");
        WorldModelCell[i][k].Frequence=0;
      }
      else if(WorldModelCell[i][k].Info==OCCUPIED)
        fprintf(fp," #  ");
      else
        fprintf(fp,"    ");
    }
    fprintf(fp,"\n");
  }
} /* DrawMap */


/*..............................................................
main() - main routine with the loops for the real ans
hypothetical steps.
..............................................................*/
main()
{
```

```
int HypotCycles, Trial, TrialMax, i, k, RealIter, LastRealIter;
int  MinIter;
state HypotState[STATEDIM], StartState[STATEDIM];
FILE *fpPath, *fpMap;
extern struct worldcell WorldModelCell[XMAX][YMAX];
extern state RobotPos[STATEDIM];
extern int XGoal, YGoal;
void Initialise(), SetParameters(), DrawMap();
boolean Goal();
double drand48();

  Initialise();
  SetParameters(&HypotCycles,&TrialMax,StartState);

  printf("ALFA = %d\n",ALFA);
  printf("BETA = %d/1000\n",BETA);
  printf("\nWorking...\n");
  Trial = 0;
  MinIter = 10000;
  LastRealIter = 0;
  fpPath = fopen("RobotPath", "w");
  fpMap = fopen("PathMap", "w");
  do{
    Trial++;
    fprintf(fpPath,"\n   TRIAL %d\n",Trial);
    for(i=0;i<STATEDIM;i++) RobotPos[i]=StartState[i];

    RealIter=0;
    do{
      RealIter++;
      RealStep(RobotPos);             /* do a real step */
      fprintf(fpPath,"%3d %3d %3d\t",RobotPos[X],RobotPos[Y],
        RobotPos[ANG]);
      fprintf(fpPath,"  Real Step %4d\n",RealIter);

      if(Trial>1){
        /* start a cycle of hypothetical trials */
        for(k=0;k<HypotCycles;k++){
          do{
            HypotState[X]=(int)(drand48()*(double)XMAX);
            HypotState[Y]=(int)(drand48()*(double)YMAX);
          }while(WorldModelCell[HypotState[X]][HypotState[Y]]
          .Info!=EMPTY || IsGoal(HypotState));
          fprintf(fpPath,"%3d %3d    \t",HypotState[X],
            HypotState[Y]);
          fprintf(fpPath,"  Hypo Step %4d\n",k+1);
          HypotStep(HypotState);   /* do a hypothetical step */
        }
      }
    }while(!IsGoal(RobotPos));
    fprintf(fpPath,"  %4d iterations\n\n",RealIter-1);
    DrawMap(fpMap,Trial,RealIter);

    /* check for optimal path and maximum number of trials */
    if(RealIter==LastRealIter && RealIter==MinIter){
      fprintf(fpPath,"\nOptimal path found in %d trials.\n",
        Trial-1);
      printf("\nOptimal path found in %d trials.\n",Trial-1);
      fprintf(fpPath,"Need %d steps to reach goal.\n\n",
        RealIter-1);
      printf("Need %d steps to reach goal.\n\n",RealIter-1);
      break;
    }
    else if (Trial>=TrialMax){
      fprintf(fpPath,"\nCouldnt find optimal path.\n\n");
      printf("\nCouldnt find optimal path.\n\n");
      break;
    }
    if(RealIter<=MinIter)
      MinIter = RealIter;
    LastRealIter = RealIter;

  }while(TRUE);

  fclose(fpPath);
  fclose(fpMap);

  MakeTables();

} /* main */
```

A.6 Preliminary File Movements.c

```
/*..............................................................
Movements.c - contains the routines to move the platform. The
routines are used by the DynAlg. This module is used to run
DynAlg in simulation modus without using the robot. Therefore
the functions InitMovements and TermMovements are empty, and
the motion process doesnt exist.
................................................................
Unit:
File:
Date:
Author:
Version:
Revisions:
Comments:

Functions:
  InitMovements()
  Rotate(Angle)
  Translate()
  TermMovements()
  GenerateSonarAlarm(State, Action)
..............................................................*/

/*..............................................................
InitMovements() - initialises and starts the motion process.
..............................................................*/
void InitMovements()
{
/* This routine and the mtoion process are not used on
    simulation */
}

/*..............................................................
Rotate(Angle)- rotates the robot to the propper
 orientation in order to make the new translation.
..............................................................*/
void Rotate(Angle)
int Angle;
{

  if (Angle!=0){
    /* executar a rotacao */
  }
} /* Rotate */

/*..............................................................
Translate()- executes the movement of the robot one step
forward.  Returns a boolean value corresponding to the
possibilty to execute the action, e.g., no obstacle.
..............................................................*/
boolean Translate()
{
boolean PositionReached=TRUE;
extern boolean SonarAlarm;

    if(SonarAlarm){
      /* return to initial position */
      PositionReached = FALSE;
      SonarAlarm = FALSE;
    }
  if(PositionReached)
    return(TRUE);
  else
    return(FALSE);
} /* Translate */

/*.................................................................
TermMovements() - remove motion process.
...............................................................*/
void TermMovements()
{

}

/*.................................................................
GenerateSonarAlarm(State,Action)- generates a sonar alarm for
simulation.
...............................................................*/
void GenerateSonarAlarm(State,Action)
pstate State;
action Action;
{
int i,x,y;

  /* left border */
  for(i=1;i<(YMAX-1);i++)
   if(State[X]==1 && State[Y]==i && Action==LEFT)
     SonarAlarm=TRUE;

  /* right border */
  for(i=1;i<(YMAX-1);i++)
   if(State[X]==(XMAX-2) && State[Y]==i && Action==RIGHT)
     SonarAlarm=TRUE;

  /* upper border */
  for(i=1;i<(XMAX-1);i++)
   if(State[X]==i && State[Y]==1 && Action==UP)
     SonarAlarm=TRUE;

  /* lower border */
  for(i=1;i<(XMAX-1);i++)
   if(State[X]==i && State[Y]==(YMAX-2) && Action==DOWN)
     SonarAlarm=TRUE;

  /* obstacle from (3,1) to (3,3) */
  /*
  for(i=1;i<4;i++)
   if(State[X]==2 && State[Y]==i && Action==RIGHT)
     SonarAlarm=TRUE;A
  for(i=1;i<4;i++)
    if(State[X]==4 && State[Y]==i && Action==LEFT)
      SonarAlarm=TRUE;
  if(State[X]==3 && State[Y]==4 && Action==UP)
    SonarAlarm=TRUE;
  */

  /* obstacle at */ x = 2; y = 3;
  if(State[X]==(x-1) && State[Y]==y && Action==RIGHT)
    SonarAlarm=TRUE;
  if(State[X]==x && State[Y]==(y-1) && Action==DOWN)
    SonarAlarm=TRUE;
```

```
if(State[X]==x && State[Y]==(y+1) && Action==UP)
  SonarAlarm=TRUE;
if(State[X]==(x+1) && State[Y]==y && Action==LEFT)
  SonarAlarm=TRUE;

/* obstacle at */ x = 3; y = 3;
if(State[X]==(x-1) && State[Y]==y && Action==RIGHT)
  SonarAlarm=TRUE;
if(State[X]==x && State[Y]==(y-1) && Action==DOWN)
  SonarAlarm=TRUE;
if(State[X]==x && State[Y]==(y+1) && Action==UP)
  SonarAlarm=TRUE;
if(State[X]==(x+1) && State[Y]==y && Action==LEFT)
  SonarAlarm=TRUE;

/* obstacle at */ x = 4; y = 3;
if(State[X]==(x-1) && State[Y]==y && Action==RIGHT)
  SonarAlarm=TRUE;
if(State[X]==x && State[Y]==(y-1) && Action==DOWN)
  SonarAlarm=TRUE;
if(State[X]==x && State[Y]==(y+1) && Action==UP)
  SonarAlarm=TRUE;
if(State[X]==(x+1) && State[Y]==y && Action==LEFT)
  SonarAlarm=TRUE;

/* obstacle at */ x = 4; y = 4;
if(State[X]==(x-1) && State[Y]==y && Action==RIGHT)
  SonarAlarm=TRUE;
if(State[X]==x && State[Y]==(y-1) && Action==DOWN)
  SonarAlarm=TRUE;
if(State[X]==x && State[Y]==(y+1) && Action==UP)
  SonarAlarm=TRUE;
if(State[X]==(x+1) && State[Y]==y && Action==LEFT)
  SonarAlarm=TRUE;

/* obstacle at */ x = 4; y = 5;
if(State[X]==(x-1) && State[Y]==y && Action==RIGHT)
  SonarAlarm=TRUE;
if(State[X]==x && State[Y]==(y-1) && Action==DOWN)
  SonarAlarm=TRUE;
if(State[X]==x && State[Y]==(y+1) && Action==UP)
  SonarAlarm=TRUE;
if(State[X]==(x+1) && State[Y]==y && Action==LEFT)
  SonarAlarm=TRUE;

/* obstacle at */ x = 4; y = 6;
if(State[X]==(x-1) && State[Y]==y && Action==RIGHT)
  SonarAlarm=TRUE;
if(State[X]==x && State[Y]==(y-1) && Action==DOWN)
  SonarAlarm=TRUE;
if(State[X]==x && State[Y]==(y+1) && Action==UP)
  SonarAlarm=TRUE;
if(State[X]==(x+1) && State[Y]==y && Action==LEFT)
  SonarAlarm=TRUE;

/* obstacle at */ x = 3; y = 6;
if(State[X]==(x-1) && State[Y]==y && Action==RIGHT)
  SonarAlarm=TRUE;
if(State[X]==x && State[Y]==(y-1) && Action==DOWN)
  SonarAlarm=TRUE;
if(State[X]==x && State[Y]==(y+1) && Action==UP)
  SonarAlarm=TRUE;
if(State[X]==(x+1) && State[Y]==y && Action==LEFT)
  SonarAlarm=TRUE;

/* obstacle at */ x = 5; y = 4;
if(State[X]==(x-1) && State[Y]==y && Action==RIGHT)
  SonarAlarm=TRUE;
if(State[X]==x && State[Y]==(y-1) && Action==DOWN)
  SonarAlarm=TRUE;
if(State[X]==x && State[Y]==(y+1) && Action==UP)
  SonarAlarm=TRUE;
if(State[X]==(x+1) && State[Y]==y && Action==LEFT)
  SonarAlarm=TRUE;

/* obstacle at */ x = 6; y = 4;
if(State[X]==(x-1) && State[Y]==y && Action==RIGHT)
  SonarAlarm=TRUE;
if(State[X]==x && State[Y]==(y-1) && Action==DOWN)
  SonarAlarm=TRUE;
if(State[X]==x && State[Y]==(y+1) && Action==UP)
  SonarAlarm=TRUE;
if(State[X]==(x+1) && State[Y]==y && Action==LEFT)
  SonarAlarm=TRUE;

/* obstacle at */ x = 7; y = 4;
if(State[X]==(x-1) && State[Y]==y && Action==RIGHT)
  SonarAlarm=TRUE;
if(State[X]==x && State[Y]==(y-1) && Action==DOWN)
  SonarAlarm=TRUE;
if(State[X]==x && State[Y]==(y+1) && Action==UP)
  SonarAlarm=TRUE;
if(State[X]==(x+1) && State[Y]==y && Action==LEFT)
  SonarAlarm=TRUE;

} /* GenerateSonarAlarm */
```

Appendix B

Results of the Simulation

B.1 Maps

B.1.1 Experiment with $\alpha = 1000$, Seed=1 and No Hypothetical Steps

```
  REAL EXPERIENCE 1 - 395 steps



        #   #
#   8  23  14   #   #       #   1
#  19  19  18   #   8   1   4   1
#  31  18   #   #   5   3   3
#  21  25  13  12   8   7
#  18  33  27  22  18  10   5
    #   #   #   #   #       #

  REAL EXPERIENCE 2 - 399 steps
    #   #   #   #   #   #   #
#  27  32  36  14  14  18  11   7   #
#  32  34  25   7  11  10  15  14   #
#  14   #   #   #  13  13   8   7   #
#   8   5   1   #   #   #   #   1
#   4   3       #
#   6   6   #   #
#   1   2
#
    #   #   #   #   #       #

  REAL EXPERIENCE 3 - 106 steps
    #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#   2   3   6   #   #   #   #   1
#   7   6  16   #  10   7   4   1
#   4   5   #   #   9   5   3   1
#   2   1   2   1   2       1   1
#   1   1   2   2
    #   #   #   #   #       #

  REAL EXPERIENCE 4 - 460 steps
    #   #   #   #   #   #   #
#  13  16   9   5   2               #
#  17  11   5                       #
#  15   #   #   #                   #
#  13  14  28   #   #   #   #   1
#  12  18  21   #   1   5   4   3
#  23  16   #   #   1   7   8   3
#  34  24  10  11   7   4   5   6   #
#  30  27  13   5   7   3   2   1
    #   #   #   #   #       #


  REAL EXPERIENCE 6 - 242 steps
    #   #   #   #   #   #   #
#                                   #
#                                   #
#   1   #   #   #                   #
#  10  16  14   #   #   #   #   1
#  10  13  13   #   5   7   2   1
#  13  15   #   #   7   5   1
#   9  11   7  10   8   8   4   4   #
#   3   7   7   8   8   9   3   2   #
    #   #   #   #   #   #   #

  REAL EXPERIENCE 7 - 96 steps
    #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#   1   1       #   1   1   2   1
#   1   1   #   #   3   2   2
#   2   2   5  10   7   2   1       #
#   8   3   8  12  10   6   3       #
    #   #   #   #   #   #   #

  REAL EXPERIENCE 9 - 296 steps
    #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#   7  17   8   #   #   #   #   1
#  21  32  22   #               1
#  22  24   #   #           1   2
#  15  18  13   3   1   1   3   5   #
#  16  13  15   4   4   7   7  13   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 10 - 37 steps
    #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#   1           #   #   #   #   1
#   4   1       #           1   1
#       1   #   #   3   2   1
#       2   2   2   2   1           #
#       4   4   2   2               #
    #   #   #   #   #   #   #   #


  REAL EXPERIENCE 11 - 471 steps
    #   #   #   #   #   #   #
#  14  14  15   7                   #
#  21  18  12   4   1               #
#  30   #   #   #   2   1   1   1   #
#  22  20  23   #   #   #   #   1
#  27  23  21   #   1   1
#  22   6   #   #   6   3
#  17  17  12   8  13   7   6   4   #
#   2  11  12  11  17  10   6   1   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 12 - 340 steps
    #   #   #   #   #   #   #
#  12   6   5                       #
#   6   4                           #
#   6   #   #   #                   #
#  20  17  20   #   #   #   #   1
#  21  17  36   #   2   4   1   1
#  26  16   #   #   4   3   3   4   #
#  15  11  17  19   5               #
#   5   5  11  13   4               #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 13 - 136 steps
    #   #   #   #   #   #   #
#   3   5   4   4   2   6   2   1   #
#   7   5   3   3   3   4   3   3   #
#   5   #   #   #   1   2   2   2   #
#  14  14  14   #   #   #   #   1
#   6   8   7   #
#       2   #   #                   #
#                                   #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 17 - 33 steps
    #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#   3   1       #               1   #
#   2   1   #   #   1       1   1   #
#       1   3   4   4   2   2       #
#                   1   2   2       #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 18 - 371 steps
    #   #   #   #   #   #   #
#  26  15  14   6   2       2       #
#  22  21  17   7   6   3   3       #
#  18   #   #   #   5   3   3   1   #
#  20  12  15   #   #   #   #   1
#  17  14  15   #                   #
#  15   8   #   #                   #
#  20  12   7   2                   #
#  16  13   5   4   1               #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 20 - 75 steps
    #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#   4  13   7   #   #   #   #   1
#   2   6   9   #   1   1   1   1   #
#   1   3   #   #   1               #
#       1   5   9   2               #
#               5   2               #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 21 - 414 steps
    #   #   #   #   #   #   #   #
#   8  11   7   6   5   4   6   8   #
#  12  30  23  10   6   8   3   2   #
#  15   #   #   #   4   7   1   1   #
#  17  22  23   #   #   #   #   1
#  13  12  20   #                   #
#  12  15   #   #                   #
#   9  18  14   8   3               #
#  14  16   7   5   7   1           #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 23 - 85 steps
    #   #   #   #   #   #   #   #
#   2   6   3   4               1   #
#   2   4   2   4   3   3   1   2   #
#   1   #   #   #               1   #
#   3  10   9   #   #   #   #   1
#       4  12   #                   #
#   3   4   #   #                   #
#                                   #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 26 - 168 steps
    #   #   #   #   #   #   #   #
#   2   2   5   2   1               #
#   7   1   1   1   1   3           #
#  11   #   #   #   7   7   1   1   #
#   9  15  19   #   #   #   #   1
#   5  14  15   #                   #
#   6  10   #   #                   #
#   3   6   1                       #
#   6   5                           #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 27 - 112 steps
    #   #   #   #   #   #   #   #
#   9   3   1                       #
#   5   5   1                       #
#   3   #   #   #                   #
#   5   8  14   #   #   #   #   1
#   6   9  12   #           1   1   #
#   1   2   #   #           1   2   #
#       1   1           1   2   6   #
#           1   3   2   1       4   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 28 - 144 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#   2   3   3   #   #   #   #   1
#   9   4   5   #   7           1   #
#   1   2   #   #  17   5   1   1   #
#   3   9  10   5   7   6   3       #
#   1   7   9   7   4   7   4       #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 29 - 155 steps
    #   #   #   #   #   #   #   #
#   2   4   5   7   2   3   1       #
#   5   7   4   8   7   4           #
#   6   #   #   #   9   4   1   1   #
#   4   3  12   #   #   #   #   1
#   5  11  17   #                   #
#   4   4   #   #                   #
#   2   3   1                       #
#   3   4   1                       #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 31 - 34 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #   1   1   1   1   #
#   1   2   #   #   3               #
#       1   3   4   2               #
#           1   6   5   1           #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 32 - 76 steps
    #   #   #   #   #   #   #   #
#   8   4   9   6   1   1   1       #
#   9   3   7   3   1   1   1       #
#   9   #   #   #           1   1   #
#   6   2       #   #   #   #   1
#   1           #                   #
#           #   #                   #
#                                   #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 33 - 80 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#   3   2   3   #   3   1   1   1   #
#   6   4   #   #   1               #
#  10  10   5   1   1               #
#  21   5   1                       #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 34 - 114 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#   3   2       #               1   #
#   9   3   #   #   2       1   1   #
#  11   7   5   6   6   5   6   3   #
#  11  12   8   4   2   2   3       #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 35 - 193 steps
    #   #   #   #   #   #   #   #
#   6  11   9  10   4   3           #
#   5   8   5   6   4   4   1       #
#   3   #   #   #   1   2   2   1   #
#   7  11  15   #   #   #   #   1
#  22  17  14   #                   #
#  12   9   #   #                   #
#                                   #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 36 - 436 steps
    #   #   #   #   #   #   #   #
#  18   4   4   1                   #
#  16   9   3   1                   #
#  11   #   #   #                   #
#  16  21  33   #   #   #   #   1
#  11  11   9   #   3   1       1   #
#  10  11   #   #   6   2   1   1   #
#  21  27  21  26  16   9   4   1   #
#  17  24  14  16  12  14   7   2   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 37 - 83 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #   2       1   1   #
#   1   8   #   #   5   3   1       #
#   1  10   9   7   2   3   1       #
#       3   6   5   2   4   6   1   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 40 - 73 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#   1   #   #   #                   #
#   3   3   1   #   #   #   #   1
#   3   5   2   #           1   1   #
#       5   #   #           1       #
#   2   5  10   8   2   1   1       #
#   2   4   6   5                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 41 - 161 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#   3   #   #   #                   #
#   9  18  15   #   #   #   #   1
#   6  19  14   #   1   1   1   1   #
#   5   6   #   #   4   3   2       #
#   4  10   8   1   1   1   1       #
#   9   8   8   1                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 42 - 60 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#   1   1   1   #   #   #   #   1
#   1       3   #           1   1   #
#   1   1   #   #       1   1       #
#   5   4   4   1   2   2           #
#   3   5   7   5   6   2           #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 45 - 30 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#   1           #           1   1   #
#   5   4   #   #   3   1   1       #
#   2   3   1   2   2   1   1       #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 46 - 260 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#  10  15  19   #   #   #   #   1
#  19  13  11   #           1   1   #
#  21  16   #   #   1   1   2       #
#  20  22  16   7   3       1       #
#  14  23  12   6   5               #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 47 - 35 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#       1   1   #   #   #   #   1
#       2   3   #               1   #
#   5   2   #   #   3   1   1   1   #
#   4   2   1   1   2               #
#   2               1               #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 48 - 270 steps
    #   #   #   #   #   #   #   #
#  33  34  23  18   4   7   3       #
#  22  23  15  14  10   6   2       #
#  10   #   #   #   8   2   1   1   #
#   6   9  12   #   #   #   #   1
#       2   3   #                   #
#       1   #   #                   #
#                                   #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 49 - 29 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#   1   2       #   4   2   1   1   #
#       3   #   #   1   1           #
#   2   3   2   1   1   1           #
#               1   1               #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 51 - 85 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#   2   1       #               1   #
#   2   2   #   #               1   #
#   2   4  16  13   1       1   1   #
#   1   4   9  10   3   1   3   6   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 53 - 45 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#       1       #   1   1   1   1   #
#   1   3   #   #   1               #
#   5   7   1   1   2   1           #
#   5  10   2                       #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 54 - 115 steps
    #   #   #   #   #   #   #   #
#  21  18  12   5                   #
#   8  15   8   3   1               #
#   1   #   #   #   1   1   1   1   #
#   1   1   3   #   #   #   #   1
#   3   2   4   #                   #
#   2   2   #   #                   #
#                                   #
#                                   #
    #   #   #   #   #   #   #   #
  REAL EXPERIENCE 55 - 219 steps
    #   #   #   #   #   #   #   #
#   3   2                           #
#   6   2                           #
#   8   #   #   #                   #
#   7  16   5   #   #   #   #   1
#   7  17  12   #               1   #
#  23  15   #   #   1           1   #
#  22  12   9   2   2           1   #
#  21   7   5   4   3   1   1   2   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 57 - 98 steps
    #   #   #   #   #   #   #   #
#       1   1   3   6   4   1       #
#   2   5   5   2   1   1   1       #
#   4   #   #   #           1   1   #
#   1   4   7   #   #   #   #   1
#   2   3   5   #                   #
#   8   4   #   #                   #
#   2   3   1                       #
#   3   5  10                       #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 58 - 95 steps
    #   #   #   #   #   #   #   #
#   6   6                           #
#   8  12   6   1   1   1   1       #
#   8   #   #   #           1   1   #
#  10   6   4   #   #   #   #   1
#   4   2   4   #                   #
#   6   5   #   #                   #
#       1                           #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 59 - 152 steps
    #   #   #   #   #   #   #   #
#   1   7   4   4   3   3           #
#   2   4   9  12   4   4   1       #
#   2   #   #   #   7   4   1   1   #
#   6   7   6   #   #   #   #   1
#  16  12   3   #                   #
#  12   7   #   #                   #
#   6                               #
#   3                               #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 60 - 302 steps
    #   #   #   #   #   #   #   #
#  11   8  20   8   6   1           #
#   8   6   9   4   2               #
#  10   #   #   #                   #
#   8   3       #   #   #   #   1
#   6   5   6   #           1   1   #
#  11  13   #   #   1       1       #
#  20  22  23   9   4   3   1       #
#   8  13  19  13   9   6   2       #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 62 - 294 steps
    #   #   #   #   #   #   #   #
#   9   5   1                       #
#   6   1                           #
#  10   #   #   #                   #
#  13  12   9   #   #   #   #   1
#  16  20  13   #           1   1   #
#  16  27   #   #       1   2       #
#  18  33  18   5   1   4   5   1   #
#   6  16  12   5   3   1   1   1   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 63 - 51 steps
    #   #   #   #   #   #   #   #
#   8   7   7   4   1               #
#   5   6   3       1   1           #
#   2   #   #   #       1   1   1   #
#   1           #   #   #   #   1
#   1           #                   #
#           #   #                   #
#                                   #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 64 - 51 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#           3   #   #   #   #   1
#       1   6   #           1   1   #
#       1   #   #           1       #
#   1   3   2   2   2   2   1       #
#   6   5   5   5   1   1           #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 65 - 101 steps
    #   #   #   #   #   #   #   #
#  14  13   5                       #
#   5   7   4   1   2   1           #
#  11   #   #   #   1   1   1   1   #
#  12   6   2   #   #   #   #   1
#   8   4       #                   #
#   1       #   #                   #
#                                   #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 66 - 27 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #           1   1   #
#       3   #   #           1       #
#   4   1   2   2   1       1       #
#   3   1   2       1   1   1       #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 67 - 273 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#   6  21  21   #   #   #   #   1
#  21  24  20   #           1   1   #
#  21  17   #   #   1   2   1       #
#   7  19  21  10   3   1           #
#   2  16  19  10   6   1           #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 68 - 72 steps
    #   #   #   #   #   #   #   #
#   2               5   4           #
#   5   1   1   3   6   2           #
#   4   #   #   #   3   1   1   1   #
#  13   7   1   #   #   #   #   1
#   4   1       #                   #
#   4   1   #   #                   #
#   1                               #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 70 - 82 steps
    #   #   #   #   #   #   #   #
#   5   4   3   3   3               #
#   5   2       1   3   1           #
#  11   #   #   #   5   2   1   1   #
#   8   3   5   #   #   #   #   1
#   4   4   6   #                   #
#       1   #   #                   #
#                                   #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 71 - 112 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#   5  12  25   #   #   #   #   1
#   3   5  16   #           1   1   #
#   5   4   #   #       1   1       #
#   2   1   5   1       1           #
#   4   6   4   3   3   2           #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 72 - 267 steps
    #   #   #   #   #   #   #   #
#   8   5   3   2   1   2           #
#  14   6           1   4   1       #
#  25   #   #   #       2   1   1   #
#  23  14  13   #   #   #   #   1
#  23  14  20   #                   #
#  17  15   #   #                   #
#  16   8   1                       #
#  16  10                           #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 74 - 139 steps
    #   #   #   #   #   #   #   #
#  10   1   2   1   3   3           #
#   4   1   3   1   2               #
#   6   #   #   #   4   1   1   1   #
#  10  23  22   #   #   #   #   1
#   5  11  16   #                   #
#   2   5   #   #                   #
#                                   #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 75 - 350 steps
    #   #   #   #   #   #   #   #
#   2   2   3                       #
#   7   7   4   1   1   1           #
#   9   #   #   #       1   1   1   #
#  27  34  37   #   #   #   #   1
#  28  39  44   #                   #
#  20  31   #   #                   #
#  14  11   5                       #
#  13   5   1                       #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 76 - 48 steps
    #   #   #   #   #   #   #   #
#   4   4   1   4   2               #
#   5   4   2   4   1               #
#   1   #   #   #   1   1   1   1   #
#   2   2   1   #   #   #   #   1
#       1   5   #                   #
#           #   #                   #
#                                   #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 78 - 27 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#   2   #   #   #                   #
#   5   2       #   #   #   #   1
#   5           #           1   1   #
#   2       #   #           1       #
#   1   1   1   1   1   1   1       #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 79 - 78 steps
    #   #   #   #   #   #   #   #
#   4   7  11   7   5   3           #
#   5   6   7   6   3   1           #
#   3   #   #   #   1   1   1   1   #
#   2           #   #   #   #   1
#   2           #                   #
#   1       #   #                   #
#                                   #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 80 - 86 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#   1  10   8   #   #   #   #   1
#   3  10  13   #           1   1   #
#   5   3   #   #       1   1       #
#   6   4   1   1   1   1           #
#   9   5                           #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 84 - 78 steps
    #   #   #   #   #   #   #   #
#  12   2                           #
#  10   2   1   1   1   1   1   1   #
#   6   #   #   #               1   #
#   3   2   3   #   #   #   #   1
#   5   4   2   #                   #
#  12   4   #   #                   #
#   3                               #
#                                   #
    #   #   #   #   #   #   #   #
```

```
  REAL EXPERIENCE 85 - 57 steps
    #   #   #   #   #   #   #   #
#   6   4   4   1           2       #
#   4   7   5   7   1   1   2   1   #
#   1   #   #   #               1   #
#   1   1       #   #   #   #   1
#       2   5   #                   #
#           #   #                   #
#                                   #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 88 - 186 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#   1   3       #   #   #   #   1
#   9  14   1   #           1   1   #
#   8  19   #   #   1   1   1       #
#  20  22  19   7   3               #
#  17  12  12   7   5   1           #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 90 - 53 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#   1   2       #           1   1   #
#   1   3   #   #   3   1   1       #
#   5   5   2   3   2               #
#   4  10   5   2                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 91 - 93 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#   3   4   4   #   #   #   #   1
#   4   7  11   #   1       1   1   #
#   5   6   #   #   2   1   1       #
#   2   3   9   4   1   1           #
#   1   2  12   4   1   1           #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 94 - 49 steps
    #   #   #   #   #   #   #   #
#   2   1   4   5   1   1           #
#   1       1   1       1           #
#   3   #   #   #       1   1   1   #
#   5           #   #   #   #   1
#   4   2   2   #                   #
#   3   1   #   #                   #
#   3   2                           #
#   1   1                           #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 96 - 15 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#       1   1   #           1   1   #
#       1   #   #   1   1   1       #
#       2   2   1   1               #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 97 - 126 steps
    #   #   #   #   #   #   #   #
#   3   2   4   2   2   2           #
#   5   3   1       1   1           #
#   2   #   #   #       1   1   1   #
#   6   9   9   #   #   #   #   1
#   9   6  10   #                   #
#   4   2   #   #                   #
#  13   6   1                       #
#  11   7   1                       #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 101 - 38 steps
    #   #   #   #   #   #   #   #
#   2   3   2   1                   #
#   4   4   2   2   1   1           #
#   1   #   #   #       1   1   1   #
#   2   6   1   #   #   #   #   1
#   1   1       #                   #
#           #   #                   #
#                                   #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 102 - 62 steps
    #   #   #   #   #   #   #   #
#   2   6   4   3   2               #
#  14   3   6   6   3   1           #
#   4   #   #   #       1   1   1   #
#   1           #   #   #   #   1
#   3           #                   #
#           #   #                   #
#                                   #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 103 - 17 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#   1           #           1   1   #
#   1       #   #   2   1   1       #
#   1   1   1   2   1               #
#   1   1                           #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 109 - 66 steps
    #   #   #   #   #   #   #   #
#       3   3   2                   #
#   1   1       1   1   1           #
#   4   #   #   #       1   1   1   #
#   1   8  11   #   #   #   #   1
#   2   7  11   #                   #
#   1   3   #   #                   #
#       1                           #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 110 - 178 steps
    #   #   #   #   #   #   #   #
#           2   1                   #
#   6   3   3   3   1   1           #
#  16   #   #   #       1   1   1   #
#  13  11   6   #   #   #   #   1
#  16  16   7   #                   #
#   6  17   #   #                   #
#   3   7   6   8   1               #
#   4   5   7   5                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 111 - 135 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#   9  17  13   #   #   #   #   1
#   2  11  14   #           1   1   #
#   8  10   #   #   1   1   1       #
#  12  14   7   2   1               #
#   2   4   3                       #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 112 - 111 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#   2   #   #   #                   #
#   8   5   1   #   #   #   #   1
#   8   9   5   #           1   1   #
#  11  15   #   #   1   1   1       #
#   9   9   3       3               #
#   3   5   3   1   5               #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 116 - 28 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#       1       #           1   1   #
```

```
#       3   #   #   1   1   1       #
#       2   3   1   1               #
#       3   7   1                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 117 - 77 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#   1   1       #           1   1   #
#   4   8   #   #   1   1   1       #
#   7   7   5   6   2   1           #
#  13   6   3   3   1   2   1       #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 118 - 40 steps
    #   #   #   #   #   #   #   #
#   5   3   1                       #
#   5   4   5   2   1   1           #
#   3   #   #   #       1   1   1   #
#   1   1       #   #   #   #   1
#       1   3   #                   #
#           #   #                   #
#                                   #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 121 - 97 steps
    #   #   #   #   #   #   #   #
#                                   #
#   1                               #
#   4   #   #   #                   #
#   5   5   8   #   #   #   #   1
#   9   6   9   #           1   1   #
#  10   5   #   #   1   1   1       #
#   5   3   5   2   1               #
#   8   1   3   1                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 122 - 45 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#   1           #   #   #   #   1
#   3           #           1   1   #
#   6   1   #   #   1   1   1       #
#   7   2           1               #
#   7   4   2   3   2               #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 123 - 70 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#       1   1   #   #   #   #   1
#   2   2   2   #               1   #
#   4   6   #   #               1   #
#  10   8   2                   2   #
#   2   6   4   2   5   2   2   4   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 126 - 108 steps
    #   #   #   #   #   #   #   #
#   4   8   2                       #
#  11  12   2   1   1   1           #
#   9   #   #   #       1   1   1   #
#   5  12  10   #   #   #   #   1
#   4  11   8   #                   #
#       3   #   #                   #
#                                   #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 127 - 82 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#       1   2   #   #   #   #   1
#   2   3   2   #           1   1   #
#   3   3   #   #   1   1   1       #
#   9  11   3   1   1               #
#   4  17  11   2   1               #
    #   #   #   #   #   #   #   #
  REAL EXPERIENCE 129 - 104 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#   3   7   4   #   #   #   #   1
#   7   8   8   #           1   1   #
#   3   9   #   #   2   1   1       #
#   5   8   6   4   3               #
#   8       5   6   3               #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 130 - 74 steps
    #   #   #   #   #   #   #   #
#  13  11   6   1                   #
#   7   6   2   1   1   1           #
#   5   #   #   #       1   1   1   #
#   4   1       #   #   #   #   1
#   1   2   6   #                   #
#       2   #   #                   #
#                                   #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 131 - 37 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#   3   #   #   #                   #
#   4   2       #   #   #   #   1
#   2           #           1   1   #
#   1       #   #           1       #
#   2   1   2   3   1   1   1       #
#   7   2   1                       #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 134 - 75 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#   4   #   #   #                   #
#   9  14   2   #   #   #   #   1
#  10  10   4   #           1   1   #
#   7   3   #   #   1   1   1       #
#   1   2   1   1   1               #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 135 - 48 steps
    #   #   #   #   #   #   #   #
#                                   #
#   4   2   1   1   1   1           #
#   9   #   #   #       1   1   1   #
#   1   4       #   #   #   #   1
#   2   1       #                   #
#   4   2   #   #                   #
#   5                               #
#   5   1                           #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 136 - 55 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#   4   8   8   #   #   #   #   1
#   9   5   7   #           1   1   #
#   3   1   #   #   1   1   1       #
#       1   1   1   1               #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 137 - 27 steps
    #   #   #   #   #   #   #   #
#       1   3                       #
#   1   1   5   2   1   1           #
#   2   #   #   #       1   1   1   #
#   1           #   #   #   #   1
#   5           #                   #
#           #   #                   #
#                                   #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 138 - 34 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#   2   4   3   #   #   #   #   1
#   1   3   1   #           1   1   #
#   1   2   #   #           1       #
#       1               1   1       #
#       1   1   3   2   2   1       #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 140 - 28 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#   4   4       #   #   #   #   1
#   1   3   5   #           1   1   #
#       1   #   #   1   1   1       #
#       1   1   1   1               #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 141 - 81 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#   4  10  22   #   #   #   #   1
#   2   7  21   #           1   1   #
#   1       #   #   1   1   1       #
#   1               1               #
#   1   1   1   2   1               #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 142 - 20 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#   1   1       #   #   #   #   1
#   1           #           1   1   #
#   1       #   #   1   1   1       #
#   3   2   1   1   1               #
#   1   1                           #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 146 - 28 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#   1   5   3   #   #   #   #   1
#       3   1   #           1   1   #
#       1   #   #   1   1   1       #
#   2   2   1   1   1               #
#   1                               #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 147 - 120 steps
    #   #   #   #   #   #   #   #
#       3   2   1                   #
#   1   1   2   2   1   1           #
#   2   #   #   #       1   1   1   #
#   9  15  23   #   #   #   #   1
#   9  13  23   #                   #
#   4   3   #   #                   #
#       1                           #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 148 - 51 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#   1   2   4   #   #   #   #   1
#   7   4   2   #           1   1   #
#   5   7   #   #   1   1   1       #
#   4   5   2   1   1               #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 149 - 23 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#       1   4   #           1   1   #
#       3   #   #   1   1   1       #
#   1   3   2   1   1               #
#       1                           #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 151 - 71 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#   3  10  15   #   #   #   #   1
#   2   6  17   #           1   1   #
#   2   1   #   #   1   1   1       #
#   1           2   1               #
#   2   1   1   1                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 152 - 46 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#   3   6   5   #   #   #   #   1
#       5   9   #           1   1   #
#   1   2   #   #   1   1   1       #
#   3   2   2   1   1               #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 155 - 15 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #           1   1   #
#       2   #   #   1   1   1       #
#   1   3   1   1   1               #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 156 - 53 steps
    #   #   #   #   #   #   #   #
#       1   1   1                   #
#   1   2       1   1   1           #
#   2   #   #           1   1   1   #
#  10   5   5   #   #   #   #   1
#  11   2   1   #                   #
#   1   3   #   #                   #
#                                   #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 158 - 22 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#       2   2   #   #   #   #   1
#       2   1   #           1   1   #
#   1   4   #   #   1   1   1       #
#       1   1   1   1               #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 159 - 29 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#   2           #   #   #   #   1
#   3   3       #           1   1   #
#       6   #   #   1   1   1       #
#   1   4   2   1   1               #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 160 - 50 steps
    #   #   #   #   #   #   #   #
#   5   1                           #
#   6                               #
#   2   #   #   #                   #
#   5   1       #   #   #   #   1
#   5   3   3   #           1   1   #
#   1   2   #   #       1   1       #
#       1   1   1   1   2           #
#                   2   3           #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 162 - 11 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #           1   1   #
#       1   #   #   1   1   1       #
#       1   1   1   1               #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 163 - 88 steps
    #   #   #   #   #   #   #   #
#  10   1       1   1               #
#   8   3   2   1   1   1           #
#  10   #   #   #       1   1   1   #
#   7   9  12   #   #   #   #   1
#   5   4   2   #                   #
#   1   4   #   #                   #
#       1                           #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 164 - 13 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #           1   1   #
#       1   #   #   1   1   1       #
#       1   3   1   1               #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 165 - 29 steps
    #   #   #   #   #   #   #   #
#                                   #
#   1   1   1   2   1   1           #
#   2   #   #   #       1   1   1   #
#   1   2   7   #   #   #   #   1
#   1   2   2   #                   #
#       1   #   #                   #
#                                   #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 166 - 18 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#       1       #           1   1   #
#       6   #   #   1   1   1       #
#       2   1   1   1               #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 167 - 51 steps
    #   #   #   #   #   #   #   #
#       6   4                       #
#   2   3   2   1   1   1           #
#  11   #   #   #       1   1   1   #
#   3           #   #   #   #   1
#   5           #                   #
#   3       #   #                   #
#   5                               #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 168 - 51 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#   3   #   #   #                   #
#   4   3   4   #   #   #   #   1
#   4   8   7   #           1   1   #
#   3   4   #   #   1   1   1       #
#   1   1   1   1   1               #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 170 - 16 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #           1   1   #
#   2   1   #   #   1   1   1       #
#   2   2   1   1   1               #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 171 - 25 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#       1       #   #   #   #   1
#       3   1   #           1   1   #
#   1   2   #   #   1   1   1       #
#   5   1   1   1   1               #
#   1   1                           #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 174 - 41 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#       6   6   #   #   #   #   1
#   3   8   1   #           1   1   #
#   1   3   #   #   1   1   1       #
#   1   2   1   1   1               #
#       1                           #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 175 - 11 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #           1   1   #
#       1   #   #   1   1   1       #
#       1   1   1   1               #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 176 - 26 steps
    #   #   #   #   #   #   #   #
#       1   1   2   2   2           #
#   1   1               1           #
#   2   #   #   #       1   1   1   #
#   4   4   1   #   #   #   #   1
#               #                   #
#           #   #                   #
#                                   #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 177 - 39 steps
    #   #   #   #   #   #   #   #
#   4   2   1                       #
#   1       1   1   1   1           #
#   5   #   #   #       1   1   1   #
#   4   3   3   #   #   #   #   1
#   4   2   1   #                   #
#   1       #   #                   #
#                                   #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 178 - 60 steps
    #   #   #   #   #   #   #   #
#                                   #
#   2   1                           #
#   4   #   #   #                   #
#   5   7   6   #   #   #   #   1
#       2   5   #           1   1   #
#       1   #   #   1   1   1       #
#   1   2   1   1   1               #
#   1   9   3   2                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 179 - 65 steps
    #   #   #   #   #   #   #   #
#                                   #
#   1   3   1   1   1   1           #
#  11   #   #   #       1   1   1   #
#   9   8   8   #   #   #   #   1
#   1   4  12   #                   #
```

```
#           #   #                   #
#                                   #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 180 - 11 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #           1   1   #
#       1   #   #   1   1   1       #
#       1   1   1   1               #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 181 - 24 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#   7   1       #           1   1   #
#   4   2   #   #   1   1   1       #
#       1   1   1   1               #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 182 - 24 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#   2           #   #   #   #   1
#   4           #           1   1   #
#   1       #   #   1   1   1       #
#   1   1   1   1   1               #
#       1   2   2   1               #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 183 - 17 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#       1       #   #   #   #   1
#   1   3       #           1   1   #
#       2   #   #   1   1   1       #
#       1   1   1   1               #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 184 - 45 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#   3   8   2   #   #   #   #   1
#   6   6   2   #           1   1   #
#   3   3   #   #   1   1   1       #
#   2   1   1   1   1               #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 185 - 20 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #           1   1   #
#   2   2   #   #   1   1   1       #
#   3   1   1   1   1               #
#   3                               #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 186 - 16 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#       2       #   #   #   #   1
#   1   2       #           1   1   #
#       1   #   #   1   1   1       #
#       1   1   1   1               #
#                                   #
    #   #   #   #   #   #   #   #
  REAL EXPERIENCE 187 - 12 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #           1   1   #
#       2   #   #   1   1   1       #
#       1   1   1   1               #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 188 - 11 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #           1   1   #
#       1   #   #   1   1   1       #
#       1   1   1   1               #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 189 - 46 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#   1   1   9   #   #   #   #   1
#   2   4  17   #           1   1   #
#       2   #   #   1   1   1       #
#       1   1   1   1               #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 190 - 18 steps
    #   #   #   #   #   #   #   #
#       2   2                       #
#   1   1   1   1   1   1           #
#   2   #   #   #       1   1   1   #
#   1   1       #   #   #   #   1
#               #                   #
#           #   #                   #
#                                   #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 192 - 13 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #           1   1   #
#       2   #   #   1   1   1       #
#       2   1   1   1               #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 193 - 45 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#   4   3   8   #   #   #   #   1
#   8   5   5   #           1   1   #
#   1   1   #   #   1   1   1       #
#       1   1   1   1               #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 194 - 11 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #           1   1   #
#       1   #   #   1   1   1       #
#       1   1   1   1               #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 195 - 11 steps
    #   #   #   #   #   #   #   #
#                                   #
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #           1   1   #
#       1   #   #   1   1   1       #
#       1   1   1   1               #
#                                   #
    #   #   #   #   #   #   #   #
```

B.1.2 Experiment with $\alpha = 1000$, Seed=1 and 10 Hypothetical Steps

```
  REAL EXPERIENCE 1 - 395 steps



        #   #
#   8  23  14   #   #       #   1
#  19  19  18   #   8   1   4   1
#  31  18   #   #   5   3   3
#  21  25  13  12   8   7
#  18  33  27  22  18  10   5
    #   #   #   #   #       #

  REAL EXPERIENCE 2 - 103 steps



#   3   #   #
#   3   5       #   #       #   1
#  10  10  11   #               1
#  10   4   #   #               1
#   5   1       4   4   2   1   1
#   7   3   1   3   4   1   2   5   #
    #   #   #   #   #       #   #

  REAL EXPERIENCE 3 - 133 steps
               #   #   #   #
           2   6   6   7   4
    1   2   6   9  13  13   5   3   #
#   5   #   #   #  14  15   5   3   #
#   1   1       #   #   #   #   1
#   1   1       #
#   5   4   #   #
#
#                                   #
    #   #   #   #   #       #   #

  REAL EXPERIENCE 4 - 63 steps
               #   #   #   #

    1                               #
#   2   #   #   #                   #
#   4   3   6   #   #   #   #   1
#   9   2   4   #       1   1   1
#   6   3   #   #       1
#   1   4   3   4   2   1
#       2   1                       #
    #   #   #   #   #       #   #

  REAL EXPERIENCE 5 - 73 steps
        #   #   #   #   #   #
    3   6  11   5   1   1
#   9   6   6   7       1   1       #
#   2   #   #   #           1   1   #
#   3   3   4   #   #   #   #   1
#           1   #
#           #   #
#
#                                   #
    #   #   #   #   #       #   #

  REAL EXPERIENCE 6 - 94 steps
    #   #   #   #   #   #   #
#   7  13   9   4   1   1   1
#   6  12   5   5   1       1       #
#   2   #   #   #           1   1   #
#   1   2   1   #   #   #   #   1
#   4   4       #
#   3   3   #   #
#   1   3   1
#                                   #
    #   #   #   #   #       #   #

  REAL EXPERIENCE 7 - 32 steps
    #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#       1   6   #   #   #   #   1
#       2   3   #   1   1   1   1
#   2   4   #   #   1
#   1   2   1   1   1
#   1   1                           #
    #   #   #   #   #       #   #

  REAL EXPERIENCE 8 - 101 steps
    #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#   1   7  31   #   #   #   #   1
#   9  12  22   #   1   1   1   1
#   5   4   #   #   1
#       1   1   1   1
#                                   #
    #   #   #   #   #       #   #

  REAL EXPERIENCE 9 - 13 steps
    #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#       1   1   #   1   1   1   1
#       1   #   #   1
#       1   1   1   1
#                                   #
    #   #   #   #   #       #   #

  REAL EXPERIENCE 10 - 11 steps
    #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #   1   1   1   1
#       1   #   #   1
#       1   1   1   1
#                                   #
    #   #   #   #   #       #   #

  REAL EXPERIENCE 11 - 11 steps
    #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #   1   1   1   1
#       1   #   #   1
#       1   1   1   1
#                                   #
    #   #   #   #   #       #   #
```

B.1.3 Experiment with $\alpha = 1000$, Seed=2 and 10 Hypothetical Steps

```
  REAL EXPERIENCE 1 - 412 steps
   #   #   #   #   #
#  16  13   9  11   8   1   2   1
#  23  20   7   5   6   4   6   4   #
#  15   #   #   #   2   1   4   1
#   6   5   3           #   #   1
    3   6   9   #   6   5   3   3
    2   5   #   #  15  11   5   4   #
        5   9   8   7  11  17   9   #
    3   8  12  12  13  10  20  17   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 2 - 70 steps
   #   #   #   #   #
#
#                                   #
#       #   #   #
#   2   4   3           #   #   1
#   3   2       #               1
    1   2   #   #               1   #
    1   3   6   3   1       3   2   #
#   3   6  11   5   1   1   4       #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 3 - 45 steps
   #   #   #   #   #
#
#   1                               #
#   2   #   #   #
#   2   4   5           #   #   1
#   1   1   3   #               1
#   2   1   #   #   1   1   1   1   #
        1   4   4   1   1   1       #
#           1   1   1   1   1       #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 4 - 47 steps
   #   #   #   #   #
#   5   3   6   3   2   1
#   2   9   7           1   1   1   #
#   2   #   #   #               1
#   1   1               #   #   1
#               #
#           #   #                   #
                                    #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 5 - 50 steps
   #   #   #   #   #
#
#                                   #
#   1   #   #   #
#   5   3   3           #   #   1
#   5   3   6   #               1
#   4   9   #   #           1   1   #
        2   1   1   1   1   1       #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 6 - 190 steps
   #   #   #   #   #
#  10   9   2
#  10  10   3   1   1               #
#  13   #   #   #   1   1   1   1
#  22  29  27   #       #   #   1
#   9  14  17   #
#   2   3   #   #                   #
    1   2                           #
#                                   #
    #   #   #   #   #   #   #   #




  REAL EXPERIENCE 7 - 121 steps
   #   #   #   #   #
#
#                                   #
#       #   #   #
#   4  10  10   #       #   #   1
#  13  13  12   #   1   1   1   1
#  11  14   #   #   1               #
#   9   6   2   1   1               #
#   9                               #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 8 - 33 steps
   #   #   #   #   #
#
#                                   #
#       #   #   #
#       2   4   #       #   #   1
#       2   3   #   1   1   1   1
#   2   8   #   #   1               #
#       3   1   1   1               #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 9 - 13 steps
   #   #   #   #   #
#
#   1   1   1   1   1               #
#   1   #   #   #   1   1   1   1
#   1   1       #       #   #   1
#               #
#           #   #                   #
#                                   #
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 10 - 13 steps
   #   #   #   #   #
#
#   1   1   1   1   1               #
#   1   #   #   #   1   1   1   1
#   1           #       #   #   1
#   1           #
#           #   #                   #
#                                   #
#                                   #
    #   #   #   #   #   #   #   #
```

B.1.4 Experiment with $\alpha = 100$, Seed=1 and 10 Hypothetical Steps

```
  REAL EXPERIENCE 1 - 395 steps


        #   #
#   8  23  14   #   #       #   1
#  19  19  18   #   8   1   4   1
#  31  18   #   #   5   3   3
#  21  25  13  12   8   7
#  18  33  27  22  18  10   5
    #   #   #   #   #       #

  REAL EXPERIENCE 2 - 84 steps


#   5   #   #
#   5   4       #   #       #   1
#   6   6       #       2   2   1
#   6   5   #   #       2   1
#   9   5   4   1   2   2
#   4   2   1   4   3   1
    #   #   #   #   #       #

  REAL EXPERIENCE 3 - 237 steps
    #
    3
    4
#  10   #   #
#  16  26  22   #   #       #   1
#  18  25  20   #           1   2
#   7  12   #   #   4   1   3   2
#   1   1   1   1   4   7  10   2
#                   5   9  14   5
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 5 - 146 steps
    #


#   7   #   #
#   6   5   4   #   #       #   1
#   5   5   3   #   2   1   1   1
#   9   8   #   #   1
#  12  15  16   4   2
#   7  14  10   5   2               #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 6 - 59 steps
    #


#       #   #
#       3       #   #       #   1
#       2   1   #   2   1   1   4   #
#   5   6   #   #   2           2
#   4   4   1   5   2
#       3   1   7   2               #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 7 - 284 steps
    #   #   #   #   #
#  13  25  27  12   8   2   1
#  20  39  26   9   4
#  19   #   #   #   3
#  10  13   6   #   #       #   1
#   4   8   7   #   2   1   1   1   #
#   2   1   #   #   1
#   2   2   3   1   1
#   6   2   1                       #
    #   #   #   #   #   #   #   #



  REAL EXPERIENCE 11 - 105 steps
    #   #   #   #   #   #   #
#   8   9   4   2   2   1   2   2
```

```
#  14   7   6   5   2   1   2   7   #
#   8   #   #   #               4   #
#   1   4   5   #   #   #   #   1
#   1   3   4   #                   #
#           #   #
#
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 13 - 37 steps
    #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #           1   1   #
#   5   6   #   #       1   1
#   3   1       2   2   1
#   1   1   2   4   4               #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 14 - 177 steps
    #   #   #   #   #   #   #
#   3   2
#   5   2                           #
#  14   #   #   #                   #
#   8  12  22   #   #   #   #   1
#  12  16  27   #   1   1   1   1   #
#  11  15   #   #   2   1
#   4   5   2   2   1
#       4   2                       #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 16 - 43 steps
    #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#           7   #   #   #   #   1
#       1   4   #           1   1   #
#       1   #   #   1   1   1       #
#   1   3   2   1   1
#  14   2                           #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 19 - 12 steps
    #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #           1   2   #
#       1   #   #           1       #
#       1   1   1   1   1   1
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 20 - 117 steps
    #   #   #   #   #   #   #   #
#  15  13   8   8   1       1   2
#   4   4   6   4   3       3   3   #
#   7   #   #   #  11   1   3   1   #
#   3           #   #   #   #   1
#   7   3   2   #                   #
#   2   1   #   #                   #
#
#                                   #
    #   #   #   #   #   #   #   #



  REAL EXPERIENCE 23 - 21 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#   1           #           1   1   #
#   3   1   #   #   1   1   1       #
#   3               1
#   1   1   2   1   1               #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 25 - 44 steps
    #   #   #   #   #   #   #   #
#
#   2                               #
#   6   #   #   #                   #
#   5   2   1   #   #   #   #   1
#   6   5   1   #           1   1   #
#   1   3   #   #       1   1       #
#       1   1   1   2   1
#                   1               #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 26 - 67 steps
    #   #   #   #   #   #   #   #
#               1   2   1
#   3   1   1   3       1   1   1   #
#   1   #   #   #               1   #
#   2   7   7   #   #   #   #   1
#  11   6   9   #                   #
#   7       #   #                   #
#
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 28 - 17 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#   1           #       1   1   1   #
#   1   1   #   #       1           #
#       1   2   3   2   1
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 29 - 32 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#       2   5   #   #   #   #   1
#       1   5   #   1   1   1   1   #
#   2   3   #   #   1               #
#   2   2   2   1   1
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 31 - 21 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#   2   1       #   1   1   1   1   #
#   2   3   #   #   1               #
#       1   1   1   1
#           2   1                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 32 - 58 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#   1   #   #   #                   #
#   2   4   5   #   #   #   #   1
#   6   7   4   #           1   1   #
#   6   4   #   #   1   1   1       #
#   3   1       1   1
#   2   2   2   1                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 34 - 49 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#   3   7   7   #   #   #   #   1
#   2   5   3   #       2   1   1   #
#   2   3   #   #       1           #
#   1   3   2   1   1   1
#                   1   1           #
```

```
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 35 - 18 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#   2           #   1   1   1   1   #
#   2   2   #   #   1               #
#       2   2   1   1
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 38 - 33 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #           1   1   #
#       1   #   #   3   2   1       #
#   1   2   1   2   2   2
#   1   5   6           1           #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 39 - 15 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #               2   #
#       4   #   #   1   1   1   1   #
#       1   1   1   1
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 40 - 35 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#       3   5   #   #   #   #   1
#   3   4   3   #           1   1   #
#   2   2   #   #   1   1   1       #
#   1   1   1       1
#           1   1   1               #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 41 - 17 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#   1           #       1   1   1   #
#   1   2   #   #   2   1           #
#       1   3   1   1
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 43 - 13 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #   2   1   1   1   #
#       1   #   #   1               #
#       1   1   2   1
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 44 - 29 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#   2   2       #   #   #   #   1
#       1       #   1   1   1   1   #
#       1   #   #   1               #
#   4   2       1   1
#   3   1   1   2   2               #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 45 - 13 steps
    #   #   #   #   #   #   #   #
```

```
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #       2   2   1   #
#       1   #   #       1           #
#       1   1   1   1   1
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 46 - 25 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#       3  10   #           1   1   #
#       2   #   #   1   1   1       #
#       1   1   1   1
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 48 - 18 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #       1   1   1   #
#       2   #   #       1           #
#   1   2   3   2   1   1
#           1                       #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 49 - 12 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #               2   #
#       1   #   #               1   #
#       1   1   1   1   1   1   1
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 51 - 18 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#       1   3   #   1   1   1   1   #
#       1   #   #   1               #
#   2   2   1   1   1
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 52 - 15 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#   1           #       1   1   1   #
#   1   1   #   #   1   1           #
#       1   1   2   1
#               1                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 53 - 12 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #   2   1   1   1   #
#       1   #   #   1               #
#       1   1   1   1
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 54 - 16 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
```

```
#       1       #   1   1   1   1   #
#       2   #   #   1               #
#       1       1   1
#       1   1   2                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 56 - 24 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#       1   2   #   1   1   1   1   #
#       1   #   #   1   1           #
#       1   2   3   2   1
#               3   1               #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 57 - 11 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #           1   1   #
#       1   #   #   1   1   1       #
#       1   1   1   1
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 58 - 18 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#       1       #   #   #   #   1
#   1   1       #   1   2   1   1   #
#   1       #   #   3               #
#   1   1   1   1   1
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 59 - 30 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#   2   #   #   #                   #
#   4   2   1   #   #   #   #   1
#   1       4   #           1   1   #
#   1       #   #   1   1   1       #
#   2   1   4   1   1
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 61 - 14 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #           1   1   #
#       1   #   #       1   1       #
#   2   2   1   1   1   1
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 62 - 18 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #           2   2   #
#       1   #   #       1   1   1   #
#       2   2   1       1
#               1   1   1           #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 65 - 17 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#       3       #   #   #   #   1
#       1       #           1   1   #
#       1   #   #       1   1       #
#       1   1   1   1   1
#                   1   1           #
```

```
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 66 - 15 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #   1   1   1   1   #
#   2   1   #   #   2               #
#   1   1   1   1   1
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 67 - 13 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #           1   1   #
#       1   #   #   1   1   1       #
#       1   2   2   1
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 68 - 17 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #               1   #
#       1   #   #               1   #
#       1   1   1               1
#               1   1   1   2   4   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 69 - 23 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#       1       #   1   1   1   1   #
#       2   #   #   1               #
#   1   4   1   1   1
#       5   1                       #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 71 - 13 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #   1   1   1   1   #
#   1   2   #   #   1               #
#       1   1   1   1
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 73 - 22 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#   3           #       1   1   1   #
#   4   3   #   #   1   1           #
#   1   1   1   2   1
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 75 - 11 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #   1   1   1   1   #
#       1   #   #   1               #
#       1   1   1   1
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 76 - 17 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#   1           #           2   1   #
#   1       #   #   1   1   2       #
#   1   1   2   2   1
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 77 - 13 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#       1       #   1   1   1   1   #
#       2   #   #   1               #
#       1   1   1   1
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 78 - 17 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #   2   2   2   1   #
#       2   #   #   1               #
#       1   1   1   2
#                   1               #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 80 - 29 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#   4   #   #   #                   #
#   4   2       #   #   #   #   1
#   2           #   1   1   1   1   #
#   2       #   #   2               #
#   1   2   2   1   2
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 81 - 28 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#       1   3   #   1   2   1   1   #
#   1   1   #   #   3   2   1       #
#   1           1   2
#   3   1   1   1                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 82 - 11 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #   1   1   1   1   #
#       1   #   #   1               #
#       1   1   1   1
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 84 - 13 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#   1           #   1   1   1   1   #
#   1   1   #   #   1               #
#       1   1   1   1
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 85 - 13 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #   1   1   1   1   #
#       1   #   #   1   1   1       #
#       1   1   1   1
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 86 - 11 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #   1   1   1   1   #
#       1   #   #   1               #
#       1   1   1   1
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 88 - 22 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#       2   3   #           1   1   #
#   1   1   #   #       2   1       #
#   1   1   1   1       2
#               1   1   1           #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 90 - 14 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #           1   1   #
#       1   #   #       1   1       #
#       1               1
#       1   2   1   1   1           #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 91 - 21 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#   1   1   1   #   #   #   #   1
#   1       1   #           1   1   #
#   4   1   #   #       1   1       #
#   1   1   1   1   1   1
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 93 - 13 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#       1       #               1   #
#       2   #   #               1   #
#       1   1   1   1   1   1   1
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 94 - 17 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#           1   #   #   #   #   1
#       1   4   #       1   1   1   #
#       1   #   #   1   1           #
#       1   1   1   1
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 96 - 15 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #   4   1   1   1   #
#       2   #   #   1               #
#       1   1   1   1
#                                   #
```

```
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 97 - 26 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#   1   1   1   #   #   #   #   1
#   2   1   2   #   1   1   1   1   #
#   1   1   #   #   2               #
#       1   1   2   2
#               1   2               #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 99 - 11 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #   1   1   1   1   #
#       1   #   #   1               #
#       1   1   1   1
#                                   #
    #   #   #   #   #   #   #   #

  REAL EXPERIENCE 100 - 13 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #       2   1   1   #
#       1   #   #       1           #
#       1   2   1   1   1
#                                   #
    #   #   #   #   #   #   #   #

REAL EXPERIENCE 102 - 14 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#       1   2   #           1   1   #
#       1   #   #   1   1   1       #
#       1   1   1   1
#                                   #
    #   #   #   #   #   #   #   #

REAL EXPERIENCE 104 - 18 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#   1           #           2   1   #
#   1       #   #       1   1       #
#   1   1   1       1   1
#           1   2   2               #
    #   #   #   #   #   #   #   #

REAL EXPERIENCE 105 - 11 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #           1   1   #
#       1   #   #           1       #
#       1   1   1   1   1   1
#                                   #
    #   #   #   #   #   #   #   #

REAL EXPERIENCE 106 - 18 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#       1       #           1   2   #
#       3   #   #   2   1   1       #
#       2   1   1   2
#                                   #
    #   #   #   #   #   #   #   #

REAL EXPERIENCE 109 - 12 steps
    #   #   #   #   #   #   #   #
```

```
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #           1   1   #
#       1   #   #   1   1   1       #
#       1   2   1   1
#                                   #
    #   #   #   #   #   #   #   #

REAL EXPERIENCE 111 - 11 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #   1   1   1   1   #
#       1   #   #   1               #
#       1   1   1   1
#                                   #
    #   #   #   #   #   #   #   #

REAL EXPERIENCE 112 - 18 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#       1       #           1   1   #
#   1   3   #   #       1   1       #
#   1   2   1   2   1   1
#                                   #
    #   #   #   #   #   #   #   #

REAL EXPERIENCE 114 - 19 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #           1   1   #
#       3   #   #       1   1       #
#   3   3   2   1   1   1
#                                   #
    #   #   #   #   #   #   #   #

REAL EXPERIENCE 115 - 11 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #           1   1   #
#       1   #   #   1   1   1       #
#       1   1   1   1
#                                   #
    #   #   #   #   #   #   #   #

REAL EXPERIENCE 116 - 26 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#   1   2       #   #   #   #   1
#   3   4   1   #           1   1   #
#   1   2   #   #       1   2       #
#       1   1   1   1   1   1
#                                   #
    #   #   #   #   #   #   #   #

REAL EXPERIENCE 118 - 11 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #           1   1   #
#       1   #   #       1   1       #
#       1   1   1   1   1
#                                   #
    #   #   #   #   #   #   #   #

REAL EXPERIENCE 119 - 22 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
```

```
#   1   1       #   6   1   1   1   #
#   2   2   #   #   1               #
#       1   2   1   1
#                                   #
    #   #   #   #   #   #   #   #

REAL EXPERIENCE 120 - 16 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #   2   1   1   1   #
#       1   #   #   3               #
#       1   1   1   1
#           1   1                   #
    #   #   #   #   #   #   #   #

REAL EXPERIENCE 123 - 11 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #           1   1   #
#       1   #   #   1   1   1       #
#       1   1   1   1
#                                   #
    #   #   #   #   #   #   #   #

REAL EXPERIENCE 124 - 11 steps
    #   #   #   #   #   #   #   #
#
#                                   #
#       #   #   #                   #
#               #   #   #   #   1
#               #           1   1   #
#       1   #   #   1   1   1       #
#       1   1   1   1
#                                   #
    #   #   #   #   #   #   #   #
```

B.1.5 Experiment with $\alpha = 1000$, Seed=1 100 Hypothetical Steps

```
  REAL EXPERIENCE 1 - 395 steps


        #   #
#   8  23  14   #   #       #   1
#  19  19  18   #   8   1   4   1
#  31  18   #   #   5   3   3
#  21  25  13  12   8   7
#  18  33  27  22  18  10   5
    #   #   #   #   #       #
  REAL EXPERIENCE 2 - 26 steps


        #   #
#       1   1   #   #       #   1
#       1   9   #   1   1   1   1
#       1   #   #   1
#       1   1   3   2
#
    #   #   #   #   #       #
  REAL EXPERIENCE 3 - 18 steps


        #   #
#               #   #       #   1
#   3   2       #       1   1   1
#       1   #   #       1
#       1               1
#       1   1   1   1   1
    #   #   #   #   #       #
  REAL EXPERIENCE 4 - 11 steps


        #   #
#               #   #       #   1
#               #   1   1   1   1
#       1   #   #   1
#       1   1   1   1
#
    #   #   #   #   #       #
  REAL EXPERIENCE 5 - 11 steps


        #   #
#               #   #       #   1
#               #   1   1   1   1
#       1   #   #   1
#       1   1   1   1
#
    #   #   #   #   #       #
```

B.1.6 Experiment with $\alpha = 1000$, Seed=5 and 100 Hypothetical Steps

```
  REAL EXPERIENCE 1 - 705 steps



#   6   #   #
#  18  18  32   #   #   #   #   1
#  26  36  36   #  10  23  17   6   #
#  37  51   #   #   9  19  18  21   #
#  22  29  26  31  19  15  10   9   #
#  20  25  26  28  17  17  11  16   #
    #   #   #   #   #   #   #   #
  REAL EXPERIENCE 2 - 28 steps

                1       1
    2   3   1   3   1   3       1
#   2   #   #   #       2   1   3   #
#   2           #   #   #   #   1
#   1           #                   #
#           #   #                   #
#                                   #
#                                   #
    #   #   #   #   #   #   #   #
  REAL EXPERIENCE 3 - 45 steps
        #   #
    1   5   4
#   2   1   1   1   1   1
#   2   #   #   #       1   1   1   #
#   4   4   5   #   #   #   #   1
#   2   2   5   #                   #
#           #   #                   #
#                                   #
#                                   #
    #   #   #   #   #   #   #   #
  REAL EXPERIENCE 4 - 13 steps
        #   #

#
#       #   #   #                   #
#               #   #   #   #   1
#               #               1   #
#       1   #   #               1   #
#       1   1   1               1   #
#               1   1   1   1   1   #
    #   #   #   #   #   #   #   #
  REAL EXPERIENCE 5 - 11 steps
        #   #

#
#       #   #   #                   #
#               #   #   #   #   1
#               #               1   #
#       1   #   #               1   #
#       1   1   1   1   1   1   1   #
#                                   #
    #   #   #   #   #   #   #   #
  REAL EXPERIENCE 6 - 11 steps
        #   #

#
#       #   #   #                   #
#               #   #   #   #   1
#               #               1   #
#       1   #   #               1   #
#       1   1   1   1   1   1   1   #
#                                   #
    #   #   #   #   #   #   #   #
```

B.1.7 Experiment with $\alpha = 100$, Seed=1 and 100 Hypothetical Steps

```
  REAL EXPERIENCE 1 - 395 steps



        #   #
#   8  23  14   #   #       #   1
#  19  19  18   #   8   1   4   1
#  31  18   #   #   5   3   3
#  21  25  13  12   8   7
#  18  33  27  22  18  10   5
    #   #   #   #   #       #

  REAL EXPERIENCE 2 - 79 steps



        #   #
#   1   3   1   #   #       #   1
#   1   3   7   #           1   1
#   2   4   #   #   1   1   1
#       5  13   9   4   1
#       6   7   5   1
    #   #   #   #   #       #

  REAL EXPERIENCE 3 - 31 steps



        #   #
#       4   3   #   #       #   1
#   1   6   4   #       1   1   1
#       2   #   #   1   1
#       1   2   1   1
#
    #   #   #   #   #       #

  REAL EXPERIENCE 4 - 17 steps



        #   #
#               #   #       #   1
#               #   1   1   2   1
#   2   1   #   #   1
#   1   1   1       1
#           1   1   1
    #   #   #   #   #       #

  REAL EXPERIENCE 5 - 68 steps
        #
    5   4
#   6   5
#   3   #   #
#   3       2   #   #       #   1
#   3   3  11   #       1   1   1
#   1   1   #   #       1
#   1                   1
#   5   4   2   1   1   1
    #   #   #   #   #       #

  REAL EXPERIENCE 6 - 13 steps
        #

#
#       #   #
#               #   #       #   1
#   1   1       #           1   1
#       1   #   #   1   1   1
#       1   1   1   1
#
    #   #   #   #   #       #




  REAL EXPERIENCE 7 - 15 steps
        #
```

```
#
#       #   #
#       1       #   #       #   1
#       2       #       1   1   1
#       2   #   #       1
#       1   1   1   1   1
#
    #   #   #   #   #       #

  REAL EXPERIENCE 8 - 13 steps
        #

#
#       #   #
#               #   #       #   1
#               #   1   1   1   1
#       1   #   #   1
#       1   1   3   1
#
    #   #   #   #   #       #

  REAL EXPERIENCE 9 - 24 steps
        #

#
#       #   #
#   3   1   1   #   #       #   1
#   1       1   #   1   1   2   1
#   1       #   #   3
#   1           1   1
#   1   1   1   1
    #   #   #   #   #       #

  REAL EXPERIENCE 10 - 19 steps
        #

#
#       #   #
#               #   #       #   1
#               #   1   1   2   1
#       3   #   #   1
#       3   3   2   1
#
    #   #   #   #   #       #

  REAL EXPERIENCE 11 - 13 steps
        #

#
#       #   #
#               #   #       #   1
#               #   1   1   1   1
#       1   #   #   1   1   1
#       1   1   1   1
#
    #   #   #   #   #       #

  REAL EXPERIENCE 12 - 14 steps
        #

#
#       #   #
#               #   #       #   1
#       1   1   #   1   1   1   1
#       1   #   #   1
#       1   1   2   1
#
    #   #   #   #   #       #




  REAL EXPERIENCE 13 - 11 steps
        #

#
#       #   #
#               #   #       #   1
#               #           1   1
#       1   #   #       1   1
#       1   1   1   1   1
#
    #   #   #   #   #       #

  REAL EXPERIENCE 14 - 13 steps
        #

#
#       #   #
#               #   #       #   1
#               #           1   1
#   1   2   #   #   1   1   1
#       1   1   1   1
#
    #   #   #   #   #       #

  REAL EXPERIENCE 15 - 14 steps
        #

#
#       #   #
#               #   #       #   1
#               #   1   1   1   1
#       1   #   #   2
#       2   2   1   1
#
    #   #   #   #   #       #

  REAL EXPERIENCE 16 - 12 steps
        #

#
#       #   #
#               #   #       #   1
#               #   1   1   1   1
#       1   #   #   2
#       1   1   1   1
#
    #   #   #   #   #       #

  REAL EXPERIENCE 17 - 11 steps
        #

#
#       #   #
#               #   #       #   1
#               #           1   1
#       1   #   #       1   1
#       1   1   1   1   1
#
    #   #   #   #   #       #

  REAL EXPERIENCE 18 - 12 steps
        #

#
#       #   #
#               #   #       #   1
#               #           1   1
#       2   #   #   1   1   1
#       1   1   1   1
#
    #   #   #   #   #       #

  REAL EXPERIENCE 19 - 13 steps
        #

#
#       #   #
#               #   #       #   1
#               #   1   1   2   1
#       1   #   #   1       1
#       1   1   1   1
#
    #   #   #   #   #       #

  REAL EXPERIENCE 20 - 33 steps
        #

#
#       #   #
#       4   4   #   #       #   1
#       2   1   #           1   1
#       1   #   #       3   3
#       1   2   1   1   2   1
#                       2   2
```

```
    #   #   #   #   #       #

  REAL EXPERIENCE 21 - 13 steps
        #

#
#       #   #
#               #   #       #   1
#               #   2   1   1   1
#       2   #   #   1
#       1   1   1   1
#
    #   #   #   #   #       #

  REAL EXPERIENCE 22 - 14 steps
        #

#
#       #   #
#               #   #       #   1
#       1       #   1   1   1   2   #
#       2   #   #   1
#       1   1   1   1
#
    #   #   #   #   #       #

  REAL EXPERIENCE 23 - 21 steps
        #

#
#       #   #
#               #   #       #   1
#               #   1   1   3   3   #
#       1   #   #   1       1   4   #
#       1   1   1   1           1
#
    #   #   #   #   #       #

  REAL EXPERIENCE 24 - 13 steps
        #

#
#       #   #
#               #   #       #   1
#               #   1   1   3   1   #
#       1   #   #   1               #
#       1   1   1   1
#
    #   #   #   #   #       #

  REAL EXPERIENCE 25 - 11 steps
        #

#
#       #   #
#               #   #       #   1
#               #           1   1   #
#       1   #   #   1   1   1       #
#       1   1   1   1
#
    #   #   #   #   #       #

  REAL EXPERIENCE 26 - 11 steps
        #

#
#       #   #
#               #   #       #   1
#               #           1   1   #
#       1   #   #   1   1   1       #
#       1   1   1   1
#
    #   #   #   #   #       #
```

B.2 Policy and Evaluation Tables

B.2.1 Experiment with $\alpha = 1000$, Seed=1 and No Hypothetical Steps

```
Evaluation Table

 0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
 0.000  0.010  0.024  0.044  0.059  0.095  0.148  0.023  0.033  0.000
 0.000  0.016  0.042  0.134  0.278  0.489  0.656  0.192  0.378  0.000
 0.000  0.007  0.000  0.000  0.000  0.131  0.785  0.895  0.999  0.000
 0.000  0.008  0.006  0.006  0.000  0.000  0.000  0.000  0.000  0.000
 0.000  0.016  0.044  0.012  0.000  0.074  0.257  0.900  1.000  0.000
 0.000  0.022  0.122  0.000  0.000  0.646  0.726  0.810  0.671  0.000
 0.000  0.066  0.231  0.354  0.463  0.559  0.193  0.356  0.226  0.000
 0.000  0.014  0.030  0.053  0.067  0.056  0.061  0.035  0.053  0.000
 0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
```

```
Policy Table

+ -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - +
|   25   |   25   |   25   |   25   |   25   |   25   |   25   |   25   |   25   |   25   |
  25   25  25   25  25   25  25   25  25   25  25   25  25   25  25   25  25   25  25   25
|   25   |   25   |   25   |   25   |   25   |   25   |   25   |   25   |   25   |   25   |
+ -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - +
|   25   |   24   |   23   |   22   |   21   |   17   |   13   |   23   |   23   |   25   |
  25   25  24   27  21   31  18   25  19   25  15   19  12   13  23   23  23   23  25   25
|   25   |   25   |   25   |   35   |   35   |   48   |   62   |   31   |   32   |   25   |
+ -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - +
|   25   |   23   |   18   |    7   |    1   |    0   |    0   |    8   |    1   |   25   |
  25   25  24   32  19   42   9   73   1   95   0   99   0    0   9   18   2    2  25   25
|   25   |   22   |   21   |   11   |    2   |    1   |   100  |   65   |   96   |   25   |
+ -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - +
|   25   |   28   |   25   |   25   |   25   |   14   |    0   |    0   |    0   |   25   |
  25   25  25   25  25   25  25   25  25   25  15   56   0  100   0  100   0    0  25   25
|   25   |   23   |   25   |   25   |   25   |   15   |    0   |    0   |   100  |   25   |
+ -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - +
|   25   |   26   |   25   |   25   |   25   |   25   |   25   |   25   |   25   |   25   |
  25   25  24   24  24   23  25   25  25   25  25   25  25   25  25   25  25   25  25   25
|   25   |   25   |   28   |   25   |   25   |   25   |   25   |   25   |   25   |   25   |
+ -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - +
|   25   |   22   |   13   |   23   |   25   |   19   |    4   |    0   |   100  |   25   |
  25   25  24   28  14   14  29   24  25   25  19   43   3   89   0  100   0    0  25   25
|   25   |   26   |   60   |   24   |   25   |   18   |    4   |    0   |    0   |   25   |
+ -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - +
|   25   |   21   |    6   |   25   |   25   |    0   |    0   |   100  |   100  |   25   |
  25   25  23   26   7   10  25   25  25   25   0  100   0  100   0    0   0    0  25   25
|   25   |   30   |   77   |   25   |   25   |    0   |    0   |    0   |    0   |   25   |
+ -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - +
|   25   |   18   |    2   |    1   |    1   |   100  |   40   |   96   |   84   |   25   |
  25   25  19   46   2   95   1   97   0   99   0    0  11   41   1    2   6    6  25   25
|   25   |   17   |    2   |    1   |    0   |    0   |    8   |    1   |    4   |   25   |
+ -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - +
|   25   |   26   |   30   |   33   |   42   |   41   |   40   |   34   |   37   |   25   |
  25   25  24   26  21   26  19   27  16   22  17   21  19   21  22   22  21   21  25   25
|   25   |   24   |   23   |   21   |   20   |   20   |   20   |   22   |   21   |   25   |
+ -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - +
|   25   |   25   |   25   |   25   |   25   |   25   |   25   |   25   |   25   |   25   |
  25   25  25   25  25   25  25   25  25   25  25   25  25   25  25   25  25   25  25   25
|   25   |   25   |   25   |   25   |   25   |   25   |   25   |   25   |   25   |   25   |
+ -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - +
```

B.2.2 Experimentwith $\alpha = 1000$, Seed=1 and 10 Hypothetical Steps

Evaluation Table

```
 0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
 0.000  0.002  0.007  0.031  0.183  0.405  0.564  0.696  0.000  0.000
 0.000  0.001  0.013  0.069  0.269  0.401  0.686  0.801  0.898  0.000
 0.000  0.002  0.000  0.000  0.000  0.706  0.798  0.896  1.000  0.000
 0.000  0.007  0.015  0.010  0.000  0.000  0.000  0.000  0.000  0.000
 0.000  0.017  0.110  0.038  0.000  0.729  0.810  0.900  1.000  0.000
 0.000  0.119  0.222  0.000  0.000  0.656  0.729  0.810  0.900  0.000
 0.000  0.191  0.337  0.450  0.521  0.589  0.655  0.729  0.810  0.000
 0.000  0.140  0.199  0.376  0.441  0.519  0.586  0.655  0.728  0.000
 0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
```

Policy Table

```
+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+
|  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |
 25  25 25  25 25  25 25  25 25  25 25  25 25  25 25  25 25  25 25  25
|  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |
+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+
|  25  |  25  |  25  |  23  |  11  |   1  |   1  |   0  |  25  |  25  |
 25  25 25  25 24  27 20  33  6  63  1  97  1  94  0   0 25  25 25  25
|  25  |  25  |  25  |  24  |  20  |   1  |   4  | 100  |  25  |  25  |
+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+
|  25  |  25  |  24  |  16  |   4  |   3  |   0  |   0  |   0  |  25  |
 25  25 25  25 23  29 14  53  3  89  2  58  0  99  0   1  0   0 25  25
|  25  |  25  |  24  |  18  |   4  |  36  |   1  |  99  | 100  |  25  |
+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+
|  25  |  25  |  25  |  25  |  25  |   0  |   0  |   0  |   0  |  25  |
 25  25 25  25 25  25 25  25 25  25  0 100  0 100  0 100  0   0 25  25
|  25  |  26  |  25  |  25  |  25  |   0  |   0  |   0  | 100  |  25  |
+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+
|  25  |  24  |  24  |  24  |  25  |  25  |  25  |  25  |  25  |  25  |
 25  25 25  25 23  24 25  24 25  25 25  25 25  25 25  25 25  25 25  25
|  25  |  26  |  29  |  27  |  25  |  25  |  25  |  25  |  25  |  25  |
+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+
|  25  |  21  |  12  |  21  |  25  |   0  |   0  |   0  | 100  |  25  |
 25  25 24  24 13  12 35  22 25  25  0 100  0 100  0 100  0   0 25  25
|  25  |  30  |  63  |  22  |  25  |   0  |   0  |   0  |   0  |  25  |
+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+
|  25  |  12  |   3  |  25  |  25  |  99  |  98  | 100  | 100  |  25  |
 25  25 17  34  3   4 25  25 25  25  0   1  0   2  0   0  0   0 25  25
|  25  |  37  |  90  |  25  |  25  |   0  |   0  |   0  |   0  |  25  |
+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+
|  25  |   6  |   1  |   1  |   0  |  80  |  99  | 100  | 100  |  25  |
 25  25  8  78  1  97  0  99  0  99  1  19  0   1  0   0  0   0 25  25
|  25  |   7  |   1  |   0  |   0  |   1  |   0  |   0  |   0  |  25  |
+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+
|  25  |  19  |  18  |   4  |  10  |   3  |  10  |  99  | 100  |  25  |
 25  25 15  50  8  64  1  93  1  86  1  95  0  89  0   0  0   0 25  25
|  25  |  15  |  11  |   2  |   2  |   1  |   1  |   0  |   0  |  25  |
+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+
|  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |
 25  25 25  25 25  25 25  25 25  25 25  25 25  25 25  25 25  25 25  25
|  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |
+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+ -   -+
```

B.2.3 Experiment with $\alpha = 1000$, Seed=2 and 10 Hypothetical Steps

```
Evaluation Table

 0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
 0.000  0.139  0.170  0.379  0.461  0.561  0.645  0.726  0.809  0.000
 0.000  0.230  0.348  0.464  0.556  0.651  0.722  0.810  0.900  0.000
 0.000  0.107  0.000  0.000  0.000  0.727  0.809  0.900  1.000  0.000
 0.000  0.055  0.004  0.001  0.000  0.000  0.000  0.000  0.000  0.000
 0.000  0.005  0.002  0.001  0.000  0.719  0.809  0.900  1.000  0.000
 0.000  0.004  0.010  0.000  0.000  0.631  0.727  0.809  0.900  0.000
 0.000  0.029  0.107  0.301  0.431  0.547  0.626  0.725  0.809  0.000
 0.000  0.009  0.053  0.167  0.325  0.424  0.521  0.640  0.726  0.000
 0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
```

```
Policy Table

+ -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - +
|   25   |   25   |   25   |   25   |   25   |   25   |   25   |   25   |   25   |   25   |
 25    25 25    25 25    25 25    25 25    25 25    25 25    25 25    25 25    25 25    25 
|   25   |   25   |   25   |   25   |   25   |   25   |   25   |   25   |   25   |   25   |
+ -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - +
|   25   |   16   |   12   |   4    |   2    |   2    |   1    |   0    |   0    |   25   |
 25    25 16    27 9     17 3     11 1     93 2     84 1     81 0     95 0      0 25    25 
|   25   |   40   |   62   |   82   |   4    |   12   |   17   |   5    |  100   |   25   |
+ -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - +
|   25   |   4    |   1    |   0    |   0    |   0    |   0    |   0    |   0    |   25   |
 25    25 5     87 1     98 0     99 0     99 0      0 0      2 0    100 0      0 25    25 
|   25   |   4    |   1    |   1    |   0    |  100   |   98   |   0    |  100   |   25   |
+ -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - +
|   25   |   58   |   25   |   25   |   25   |   0    |   0    |   0    |   0    |   25   |
 25    25 15    15 25    25 25    25 25    25 0    100 0    100 0    100 0      0 25    25 
|   25   |   11   |   25   |   25   |   25   |   0    |   0    |   0    |  100   |   25   |
+ -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - +
|   25   |   40   |   25   |   25   |   25   |   25   |   25   |   25   |   25   |   25   |
 25    25 21    20 26    25 25    25 25    25 25    25 25    25 25    25 25    25 25    25 
|   25   |   20   |   24   |   25   |   25   |   25   |   25   |   25   |   25   |   25   |
+ -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - +
|   25   |   26   |   25   |   25   |   25   |   0    |   0    |   0    |  100   |   25   |
 25    25 25    24 25    25 25    25 25    25 0    100 0    100 0    100 0      0 25    25 
|   25   |   25   |   25   |   25   |   25   |   0    |   0    |   0    |   0    |   25   |
+ -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - +
|   25   |   25   |   23   |   25   |   25   |   99   |   1    |   0    |  100   |   25   |
 25    25 25    25 23    24 25    25 25    25 0      1 0     98 0    100 0      0 25    25 
|   25   |   25   |   29   |   25   |   25   |   0    |   0    |   0    |   0    |   25   |
+ -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - +
|   25   |   23   |   10   |   2    |   1    |   94   |   0    |  100   |  100   |   25   |
 25    25 23    31 11    66 1     96 0     99 1      5 0     99 0      0 0      0 25    25 
|   25   |   23   |   12   |   1    |   0    |   1    |   0    |   0    |   0    |   25   |
+ -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - +
|   25   |   25   |   28   |   12   |   11   |   84   |   93   |   95   |  100   |   25   |
 25    25 24    26 20    30 5     74 2     83 2     11 1      4 0      4 0      0 25    25 
|   25   |   24   |   22   |   9    |   4    |   3    |   1    |   1    |   0    |   25   |
+ -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - +
|   25   |   25   |   25   |   25   |   25   |   25   |   25   |   25   |   25   |   25   |
 25    25 25    25 25    25 25    25 25    25 25    25 25    25 25    25 25    25 25    25 
|   25   |   25   |   25   |   25   |   25   |   25   |   25   |   25   |   25   |   25   |
+ -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - + -    - +
```

B.2.4 Experiment with $\alpha = 100$, Seed=1 and 10 Hypothetical Steps

Evaluation Table

0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.259	0.316	0.382	0.444	0.530	0.606	0.698	0.788	0.000
0.000	0.223	0.337	0.399	0.517	0.611	0.687	0.796	0.888	0.000
0.000	0.184	0.000	0.000	0.000	0.605	0.738	0.867	0.995	0.000
0.000	0.184	0.203	0.181	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.218	0.269	0.223	0.000	0.711	0.796	0.881	1.000	0.000
0.000	0.266	0.330	0.000	0.000	0.627	0.707	0.769	0.892	0.000
0.000	0.322	0.394	0.449	0.505	0.561	0.635	0.694	0.798	0.000
0.000	0.291	0.337	0.369	0.431	0.482	0.543	0.624	0.704	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Policy Table

```
+ -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - +
|  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |
 25  25 25  25 25  25 25  25 25  25 25  25 25  25 25  25 25  25 25  25
|  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |
+ -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - +
|  25  |  22  |  20  |  18  |  16  |  15  |  15  |  16  |  11  |  25  |
 25  25 22  38 12  47  8  56  6  63  7  63  6  64  6  36  4  11 25  25
|  25  |  17  |  20  |  18  |  15  |  15  |  15  |  42  |  75  |  25  |
+ -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - +
|  25  |  21  |  14  |  10  |   8  |   7  |   4  |   4  |   3  |  25  |
 25  25 23  38 10  57  8  67  7  71  6  78  3  89  3  87  2   6 25  25
|  25  |  18  |  18  |  15  |  13  |   9  |   4  |   6  |  89  |  25  |
+ -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - +
|  25  |  25  |  25  |  25  |  25  |   9  |  13  |   8  |   2  |  25  |
 25  25 24  24 25  25 25  25 25  25 17  58  6  68  4  78  3   6 25  25
|  25  |  28  |  25  |  25  |  25  |  17  |  13  |  10  |  89  |  25  |
+ -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - +
|  25  |  16  |  22  |  24  |  25  |  25  |  25  |  25  |  25  |  25  |
 25  25 23  22 17  16 24  24 25  25 25  25 25  25 25  25 25  25 25  25
|  25  |  40  |  45  |  29  |  25  |  25  |  25  |  25  |  25  |  25  |
+ -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - +
|  25  |  12  |   8  |  12  |  25  |  10  |   6  |   5  |  93  |  25  |
 25  25 21  21 10   8 46  21 25  25 10  77  2  90  1  92  1   4 25  25
|  25  |  46  |  74  |  21  |  25  |   3  |   2  |   2  |   1  |  25  |
+ -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - +
|  25  |  10  |   4  |  25  |  25  |  51  |  15  |  91  |  91  |  25  |
 25  25 21  32  6  11 25  25 25  25 15  30  4  78  2   5  2   5 25  25
|  25  |  37  |  78  |  25  |  25  |   4  |   3  |   2  |   2  |  25  |
+ -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - +
|  25  |   8  |   5  |  10  |   8  |  73  |  79  |  84  |  88  |  25  |
 25  25 17  59  5  83  3  82  3  85  4  19  4  13  3  10  3   7 25  25
|  25  |  15  |   7  |   5  |   4  |   4  |   4  |   3  |   3  |  25  |
+ -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - +
|  25  |  13  |  14  |  17  |  20  |  46  |  56  |  56  |  75  |  25  |
 25  25 21  46  8  61  7  60  7  56  5  32  6  22  5  23  3  11 25  25
|  25  |  21  |  17  |  17  |  17  |  17  |  16  |  16  |  11  |  25  |
+ -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - +
|  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |
 25  25 25  25 25  25 25  25 25  25 25  25 25  25 25  25 25  25 25  25
|  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |
+ -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - +
```

B.2.5 Experiment with $\alpha = 1000$, Seed=1 100 Hypothetical Steps

Evaluation Table

0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.257	0.293	0.243	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.303	0.340	0.269	0.000	0.729	0.810	0.900	1.000	0.000
0.000	0.342	0.384	0.000	0.000	0.656	0.729	0.810	0.000	0.000
0.000	0.378	0.429	0.478	0.531	0.590	0.656	0.000	0.000	0.000
0.000	0.345	0.385	0.429	0.477	0.531	0.590	0.531	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Policy Table

```
+ - - - + - - - + - - - + - - - + - - - + - - - + - - - + - - - + - - - + - - - +
|  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |
  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25
|  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |
+ - - - + - - - + - - - + - - - + - - - + - - - + - - - + - - - + - - - + - - - +
|  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |
  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25
|  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |
+ - - - + - - - + - - - + - - - + - - - + - - - + - - - + - - - + - - - + - - - +
|  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |
  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25
|  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |
+ - - - + - - - + - - - + - - - + - - - + - - - + - - - + - - - + - - - + - - - +
|  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |
  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25
|  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |
+ - - - + - - - + - - - + - - - + - - - + - - - + - - - + - - - + - - - + - - - +
|  25   |   9   |   2   |  10   |  25   |  25   |  25   |  25   |  25   |  25   |
  25  25   9  14   2   1  65  10  25  25  25  25  25  25  25  25  25  25  25  25
|  25   |  69   |  95   |  16   |  25   |  25   |  25   |  25   |  25   |  25   |
+ - - - + - - - + - - - + - - - + - - - + - - - + - - - + - - - + - - - + - - - +
|  25   |   3   |   1   |   2   |  25   |   0   |   0   |   0   |  100  |  25   |
  25  25   4   5   1   1  92   3  25  25   0 100   0 100   0 100   0   0  25  25
|  25   |  87   |  97   |   3   |  25   |   0   |   0   |   0   |   0   |  25   |
+ - - - + - - - + - - - + - - - + - - - + - - - + - - - + - - - + - - - + - - - +
|  25   |   2   |   1   |  25   |  25   |  99   |  92   |  100  |  25   |  25   |
  25  25   4  76   1   1  25  25  25  25   0   0   0   8   0   0  25  25  25  25
|  25   |  18   |  98   |  25   |  25   |   0   |   0   |   0   |  25   |  25   |
+ - - - + - - - + - - - + - - - + - - - + - - - + - - - + - - - + - - - + - - - +
|  25   |   1   |   0   |   0   |   0   |  99   |  100  |  25   |  25   |  25   |
  25  25   1  97   0  99   0  99   0  99   0   0   0   0  25  25  25  25  25  25
|  25   |   1   |   0   |   0   |   0   |   0   |   0   |  25   |  25   |  25   |
+ - - - + - - - + - - - + - - - + - - - + - - - + - - - + - - - + - - - + - - - +
|  25   |   2   |   2   |   8   |   1   |   6   |  100  |   0   |  25   |  25   |
  25  25   2  93   1  95   1  88   1  98   1  92   0   0  99   0  25  25  25  25
|  25   |   2   |   2   |   2   |   1   |   1   |   0   |   0   |  25   |  25   |
+ - - - + - - - + - - - + - - - + - - - + - - - + - - - + - - - + - - - + - - - +
|  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |
  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25
|  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |
+ - - - + - - - + - - - + - - - + - - - + - - - + - - - + - - - + - - - + - - - +
```

B.2.6 Experiment with $\alpha = 1000$, Seed=5 and 100 Hypothetical Steps

```
Evaluation Table

 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
 0.000 0.376 0.421 0.470 0.522 0.000 0.655 0.000 0.000 0.000
 0.000 0.425 0.474 0.528 0.588 0.654 0.729 0.000 0.900 0.000
 0.000 0.376 0.000 0.000 0.000 0.000 0.810 0.900 1.000 0.000
 0.000 0.272 0.290 0.270 0.000 0.000 0.000 0.000 0.000 0.000
 0.000 0.302 0.341 0.306 0.000 0.729 0.810 0.900 1.000 0.000
 0.000 0.339 0.380 0.000 0.000 0.656 0.729 0.810 0.900 0.000
 0.000 0.380 0.424 0.473 0.527 0.589 0.656 0.729 0.810 0.000
 0.000 0.343 0.381 0.430 0.478 0.531 0.590 0.656 0.729 0.000
 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
```

```
Policy Table

+ -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - +
|  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |
25  25 25  25 25  25 25  25 25  25 25  25 25  25 25  25 25  25 25  25
|  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |
+ -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - +
|  25  |   6  |   3  |   1  |   0  |  25  |   0  |  25  |  25  |  25  |
25  25  6  58  1  82  0  98  0   0 25  25  0   0 25  25 25  25 25  25
|  25  |  29  |  14  |   1  |  99  |  25  | 100  |  25  |  25  |  25  |
+ -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - +
|  25  |   0  |   0  |   0  |   0  |   0  |   0  |  25  |   0  |  25  |
25  25  1  99  0  99  0  99  0 100  0 100  0   0 25  25  0   0 25  25
|  25  |   0  |   0  |   0  |   0  |   0  | 100  |  25  | 100  |  25  |
+ -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - +
|  25  |  94  |  25  |  25  |  25  |  25  |   0  |   0  |   0  |  25  |
25  25  2   2 25  25 25  25 25  25 25  25  0 100  0 100  0   0 25  25
|  25  |   2  |  25  |  25  |  25  |  25  |   0  |   0  | 100  |  25  |
+ -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - +
|  25  |   7  |   2  |  10  |  25  |  25  |  25  |  25  |  25  |  25  |
25  25  7   8  1   1 26  10 25  25 25  25 25  25 25  25 25  25 25  25
|  25  |  78  |  95  |  55  |  25  |  25  |  25  |  25  |  25  |  25  |
+ -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - +
|  25  |   2  |   1  |   1  |  25  |   0  |   0  |   0  | 100  |  25  |
25  25  4   6  1   1 94   2 25  25  0 100  0 100  0 100  0   0 25  25
|  25  |  88  |  97  |   2  |  25  |   0  |   0  |   0  |   0  |  25  |
+ -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - +
|  25  |   3  |   1  |  25  |  25  |   0  |   0  |   0  | 100  |  25  |
25  25  6  59  1   1 25  25 25  25  0 100  0 100  0 100  0   0 25  25
|  25  |  33  |  97  |  25  |  25  |   0  |   0  |   0  |   0  |  25  |
+ -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - +
|  25  |   1  |   0  |   0  |   0  |   2  |   0  |   0  | 100  |  25  |
25  25  1  96  0  99  0  99  0  99  0  97  0 100  0 100  0   0 25  25
|  25  |   2  |   0  |   0  |   0  |   0  |   0  |   0  |   0  |  25  |
+ -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - +
|  25  |   2  |   1  |   1  |   1  |   0  |   2  |   0  | 100  |  25  |
25  25  3  93  1  98  1  97  1  98  0  99  0  97  0  99  0   0 25  25
|  25  |   3  |   1  |   1  |   1  |   0  |   1  |   0  |   0  |  25  |
+ -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - +
|  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |
25  25 25  25 25  25 25  25 25  25 25  25 25  25 25  25 25  25 25  25
|  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |  25  |
+ -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - + -  - +
```

B.2.7 Experiment with $\alpha = 100$, Seed=1 and 100 Hypothetical Steps

```
Evaluation Table

 0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
 0.000  0.159  0.136  0.000  0.000  0.000  0.000  0.000  0.000  0.000
 0.000  0.191  0.156  0.000  0.000  0.000  0.000  0.000  0.000  0.000
 0.000  0.215  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
 0.000  0.237  0.267  0.244  0.000  0.000  0.000  0.000  0.000  0.000
 0.000  0.263  0.295  0.268  0.000  0.712  0.796  0.889  0.996  0.000
 0.000  0.294  0.329  0.000  0.000  0.636  0.708  0.793  0.816  0.000
 0.000  0.333  0.379  0.438  0.494  0.549  0.631  0.663  0.648  0.000
 0.000  0.307  0.336  0.371  0.427  0.489  0.555  0.540  0.000  0.000
 0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
```

```
Policy Table

+ -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - +
|  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |
 25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25
|  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |
+ -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - +
|  25   |  21   |  24   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |
 25  25  21  12  26  24  25  25  25  25  25  25  25  25  25  25  25  25  25  25
|  25   |  46   |  26   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |
+ -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - +
|  25   |   8   |  12   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |
 25  25  15   8  45  21  25  25  25  25  25  25  25  25  25  25  25  25  25  25
|  25   |  69   |  21   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |
+ -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - +
|  25   |   6   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |
 25  25  17  17  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25
|  25   |  61   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |
+ -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - +
|  25   |   7   |  15   |  23   |  25   |  25   |  25   |  25   |  25   |  25   |
 25  25  18  17  8    8  21  23  25  25  25  25  25  25  25  25  25  25  25  25
|  25   |  58   |  69   |  32   |  25   |  25   |  25   |  25   |  25   |  25   |
+ -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - +
|  25   |   7   |   4   |   6   |  25   |   8   |   4   |   4   |  90   |  25   |
 25  25  18  18  4    4  61  16  25  25  8   82  1   94  1   94  1    5  25  25
|  25   |  57   |  88   |  16   |  25   |   2   |   1   |   1   |   4   |  25   |
+ -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - +
|  25   |   5   |   3   |  25   |  25   |  55   |  11   |  91   |  45   |  25   |
 25  25  18  37  4    9  25  25  25  25  13  30  2   86  1    5  26  18  25  25
|  25   |  40   |  85   |  25   |  25   |   2   |   2   |   3   |  11   |  25   |
+ -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - +
|  25   |   4   |   3   |   7   |   7   |  72   |  93   |  69   |  39   |  25   |
 25  25  11  79  2   91  2   88  2   89  2   23  2    4  13  10  20  20  25  25
|  25   |   6   |   4   |   3   |   3   |   3   |   1   |   8   |  20   |  25   |
+ -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - +
|  25   |  10   |  11   |  15   |  18   |  46   |  90   |  26   |  25   |  25   |
 25  25  19  51  4   71  4   67  4   64  3   35  2    2  31  22  25  25  25  25
|  25   |  19   |  13   |  14   |  14   |  15   |   6   |  22   |  25   |  25   |
+ -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - +
|  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |
 25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25  25
|  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |  25   |
+ -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - +
```

Appendix C

Source Code for Albatros

C.1 File README (Operating Instructions)

```
DESCRIPTION AND OPERATING INSTRUCTIONS FOR experiments.

This directory contains the files necessary to simulate
and execute on an Albatros machine the algorithm developed
in the work:

Application of an Integrated Architecture for Learning,
Planning and Reacting on a Mobile Robot.

The files are:
- DynAlg.h
- DynAlg.c
- Movements.h
- Movements.c
- Makefile

Note: Lines beginning with '>' represent the lines that appear
on the screen of your computer when you run the program.
e.g.: >kim{weiseral}1:

PART1: RUNNING A SIMULATION ON THE ALBATROS MACHINE

Before starting:

You must follow the same 4 steps mentioned in the description
of the simulationV1.0, and ...

5. Make sure that the variable of type switcher 'Simul' in the
   file DynAlg.h is set to ON.

Compile, download and start the program:

The standard communication port of the albatros machine must be
connected with the serial port A of the workstation. Type:

>kim{weiseral}5: make load

and the Makefile will compile and download the code to the
Albatros machine, that can take a few minutes.

>cc -I/home/local/asdp/include -target sun3 -c -m68020 -f68881
```

```
>-O DynAlg.c
>/usr/arch/sun4/commonR4.0/crossbin/target sun3 ld -S -u start
>-e start -N -T 20000 -o DynAlg \
>/home/local/asdp/lib/sunfp/start.o \
>DynAlg.o \
>/home/local/asdp/lib/sunfp/libcalbh.a  /home/local/asdp/lib/su
>nfp/libcalbl.a  /home/local/asdp/lib/sunfp/libcpro.a  /home/lo
>cal/asdp/lib/sunfp/libcstd.a  /home/local/asdp/lib/sunfp/libio
>.a  -lc \
>     /home/local/xcomp/sun4R4.0/sun3R4.0/usr/lib/f68881/libm.a
>  /home/local/xcomp/sun4R4.0/sun3R4.0/usr/lib/libc.a
>/home/local/asdp/bin/sformat -o DynAlg.mx DynAlg
>/home/local/asdp/bin/sload -d /dev/ttya DynAlg.mx
>kim{weiseral}6:
```

Now make a connection to the Albatros machine

```
>kim{weiseral}6: tip -9600 /dev/ttya
>connected
>
>Robuter 5.2>
```

Start the program by typing:

```
>Robuter 5.2>g 20000
```

From now on the program runs as it did in simulationV1.0.

```
>Initializing...
>
>Enter number of hypothetical cycles: 50
>Enter number of trials: 10
>Enter seed (unsigned short): 8
>Enter start state (x, y, ang): (1, 1, 270)
>
>Assign goal state (x,y): 1 3  (1,3)
>
>ALFA = 1000
>BETA = 100/1000
>
>Working...
>
>Optimal path found in 3 trials.
>Need 2 steps to reach goal.
>
>Which trial do you want to see (1-4) ?
>91: evaluation table
>92: policy table
>99: write everything in a file
>0:  quit
```

Here you can see the results in an interactive way. The information is the same described in simulationV1.0.
You can see the map of one of the trials by typing in the trial number or one of the tables by typing 91 or 92. After a quit you don't have access to the information again.
To quit type:

```
>0
>
>Robuter 5.2>
```

```
To finish connection with Albatros type '~.' (tilde point)

>Robuter 5.2>~
>[EOT]
>kim{weiseral}7:

PART2: RUNNING AN EXPERIMENT USING THE ROBOT

Before starting:

You must follow the same 4 steps mentioned in the description
of the simulationV1.0, and ...

5. Make sure that the variable of type switcher 'Simul' in the
   file DynAlg.h is set to OFF.

6. If you are simulating obstacles in addition to the real ob-
   stacles detected by the robot, set the switcher variable
   'SimObst' in DynAlg.h to ON.

Compile and  download the program:

The compilation and download is made in the same form as
described in part1.

Start the program:

Before you really start it you should establish a connection
to Albatros vi modem. Do the following:

Switch on both modems and open a new window.
Type:

>kim{weiseral}6: tip -9600 /dev/ttyb
>connected

This window is now called the window B. 'B>' indicates that the
the text appears in this window, like:

B>kim{weiseral}6: tip -9600 /dev/ttyb
B>connected

The first window used until now is window A.

Go back to window A and start the program as in part1:

A>Robuter 5.2>g 20000
A>
A>Initializing...
A>
A>Enter number of hypothetical cycles: 50
A>Enter number of trials: 10
A>Enter seed (unsigned short): 8
A>Enter start state (x, y, ang): (1, 1, 270)
A>
A>Assign goal state (x,y): 1 3  (1,3)
A>
A>ALFA = 1000
A>BETA = 100/1000
A>
A>Working...
```

```
In window B you will see:

B>Cut the link and press <Enter> ...

Now you can disconnect the cable from the serial port of the
Albatros machine. If you leave it connected you can see status
information on window A.

Press <Enter> on window B to start the robot.

The following information will appear in the windows:
(On window A only if the cable is still connected)

Robot is initialized
B>ACK     STOP    READY   INIT
A scan reading to the rear side is executed.
B>Scan back
These are the values from left, central and right sonar.
A>L 4957  C 5001  R 5043
An obstacle is detected
B>Sonar Alarm
Three new sonar readings
A>L 4963  C 5008  R 5073
Same obstacle detected again
B>Sonar Alarm
The turn right command is then aborted
B>NEW_COM RIGH    READY   OK
B>ACK     RIGH    ABORT   OBSTAC
New scan, now to the front side
B>Scan front
Six readings are made
A>L 19651 C 23006 R 19253 L 19676 C 22986  R 24147
Since the robot doesnt rotate, no angle offset
A>Offset[ANG]=0
Now it begins a translation
B>NEW_COM FORW    READY   OK
B>ACK     FORW    EXECUT  OK
This are the sonar readings during translation
A>L 19688 C 23003 R 24159 L 19688 C 22998 R 23509
A>L 19670 C 22962 R 19159 L 21402 C 22809 R 19012
A>L 21270 C 19181 R 18874 L 19338 C 22523 R 18706
A>L 19117 C 22338 R 18548 L 20706 C 22118 R 22782
  .    .      .       .       .       .       .
  .    .      .       .       .       .       .
  .    .      .       .       .       .       .
A>L 13655 C 12119 R 16211 L 13476 C 11922 R 15964
A>L 14291 C 14911 R 15747 L 14125 C 14424 R 15534
A>L 13956 C 14240 R 14043 L 13810 C 14141 R 13899
A>L 13696 C 14030 R 13780 L 13611 C 13936 R 13691
A>L 13552 C 13911 R 13628 L 13508 C 14705 R 13594
A>L 13465
Translation completed
B>ACK     STOP    READY   OK
A>Encoder 2007
Offset for the axis 2 (Y) is now 7mm
A>Offset[2]=7
dito
B>Scan back
A>L 14092 C 808   R 14219 L 14092 C 808   R 14219
Robot makes a 180 degree rotation to the right
B>NEW_COM RIGH    READY   OK
```

```
B>ACK     RIGH    EXECUT  OK
B>ACK     STOP    READY   OK
A>Encoder 181
Offset of 1 degree in the angle.
A>Offset[ANG]=-1
New translation
B>NEW_COM FORW    READY   OK
B>ACK     FORW    EXECUT  OK
A>L 11973 C 12051 R 12160 L 11955 C 12017 R 12111
A>L 11901 C 11955 R 12038 L 11777 C 11865 R 11926
  .    .      .       .       .       .       .
  .    .      .       .       .       .       .
  .    .      .       .       .       .       .
A>L 10263 C 10211 R 10185 L 10002 C 9977  R 9931
A>L 9745  C 9701  R 9640  L 9462  C 9419  R 9359
B>ACK     STOP    READY   OK
A>Encoder 2007
A>Offset[2]=0

A>8 RealIter

After reaching the goal you must bring the robot back to its
initial position. This message also appears in the middle of a
longer trial allowing to correct the odometry errors.

B>The joystick is now switched on.
B>
B>Please move the robot to the correct position
B>and press <Enter> ...

Example of an evasive action:

B>Scan left
A>L 7394  C 7418  R 7300  L 7397  C 7425  R 7440
B>NEW_COM LEFT    READY   OK
B>ACK     LEFT    EXECUT  OK
B>ACK     STOP    READY   OK
A>Encoder 89
A>Offset[ANG]=0
B>NEW_COM FORW    READY   OK
B>ACK     FORW    EXECUT  OK
A>C 5637  R 5656  L 5615  C 5625 R 5656  L 5610
A>C 5623  R 5645  L 5606  C 5604 R 5600  L 5590
  .    .      .       .       .       .       .
  .    .      .       .       .       .       .
  .    .      .       .       .       .       .
A>C 3768  R 3726  L 3579  C 3520 R 3470  L 3298
A>C 3232  R 3164  L 3007  C 2946 R 2904  L 2734
A>C 2675
The robot detects an obstacle during a translation.
B>Sonar Alarm
B>ACK     FORW    EXECUT  OBSTAC
A>Encoder 635
The translation movement is killed.
B>EM_STO  KILL    EXECUT  OBSTAC
B>ACK     KILL    EXECUT  OBSTAC
B>ACK     STOP    READY   TROUBL
A>Encoder 1014
The robot goes back to initial position
B>NEW_COM BACK    READY   TROUBL
B>ACK     BACK    EXECUT  TROUBL
B>ACK     STOP    READY   OK
```

```
A>Encoder 1028
A>Offset[1]=-14
A new action is tried
B>Scan right
A>L 24267 C 23218 R 24851 L 24271 C 23232 R 24881 Encoder 91
B>NEW_COM RIGH    READY   OK
B>ACK     RIGH    EXECUT  OK
B>ACK     STOP    READY   OK
A>Encoder 91
A>Offset[ANG]=-1
B>NEW_COM FORW    READY   OK
B>ACK     FORW    EXECUT  OK
A>R 24191 L 19668 C 19863 R 23794 L 19664 C 22910
A>R 23685 L 19623 C 22828 R 24049 L 21094  C 22648
```

At the end of a experience following message appears.

```
B>
B>The experience has been completed.
B>
B>The joystick is now switched on.
B>
B>Please move the robot to the initial position.
B>
B>To see the results, restore the link
B>and press <Enter> ...
B>
```

If you have disconnected the cable, reconnect it now and press <Enter> on window B. It continues as described in part1.

```
A>Optimal path found in 3 trials.
A>Need 2 steps to reach goal.
A>
A>
A>Which trial do you want to see (1-4) ?
A>91: evaluation table
A>92: policy table
A>99: write everything in a file
A>0:  quit
```

To make a redirection to a file, type 99:

```
A>99
```

```
B>Please finish tip connection, make a redirection
B>to a file with: cat /dev/ttya >> <filename>
B>and press <Enter> ...
```

Press <Enter> in window B.
Type in window A, use a filename of an existent file:

```
A>Robuter 5.2>~
A>[EOT]
A>kim{weiseral}13: cat /dev/ttya >> testfile
```

```
B>Please cancel redirection, restore tip connection
B>and press <Enter> ...
```

Press <Enter> in window B and do the actions.

```
A>^C
```

```
A>kim{weiseral}14: !t
A>tip -9600 /dev/ttya
A>connected

Press <Enter> in window B.

A>Which trial do you want to see (1-4) ?
A>91: evaluation table
A>92: policy table
A>99: write everything in a file
A>0:  quit
```

C.2 Makefile

```
#----------------------------------------------------------------
# Makefile for passear and later for DynAlg
#----------------------------------------------------------------
# Unit:
# File:
# Date:
# Author:
# Version:
# Revisions:
# Comments:
#
# Functions:
#----------------------------------------------------------------

#----------------------------------------------------------------
#  general variables
#----------------------------------------------------------------

ASDP_DIR=/home/local/asdp
ASDP_INC=$(ASDP_DIR)/include
ASDP_LIB=$(ASDP_DIR)/lib/sunfp
ASDP_BIN=$(ASDP_DIR)/bin
ASDP_LIB2=/home/local/xcomp/sun4R4.0/sun3R4.0/usr/lib


LD=/usr/arch/sun4/commonR4.0/crossbin/target sun3 ld
SFORMAT = $(ASDP_BIN)/sformat
SLOAD = $(ASDP_BIN)/sload

#.SUFFIXES:
#.SUFFIXES: .c.o $(SUFFIXES)
#.c.o:
# cc $(CGLAGS) $*.o

#----------------------------------------------------------------
#   compiler and linker options
#----------------------------------------------------------------

C_OPTIONS= -target sun3 -c -m68020 -f68881 -O
INCLUDES= -I$(ASDP_INC)

CFLAGS= $(INCLUDES) $(C_OPTIONS)
LDFLAGS= -S -u start -e start -N -T 20000

#----------------------------------------------------------------
#   libraries
#----------------------------------------------------------------

LLIBS= $(ASDP_LIB)/libcalbh.a \
$(ASDP_LIB)/libcalbl.a \
$(ASDP_LIB)/libcpro.a \
$(ASDP_LIB)/libcstd.a \
        $(ASDP_LIB)/libio.a \
-lc

LLIBS2= $(ASDP_LIB2)/f68881/libm.a \
        $(ASDP_LIB2)/libc.a
```

```
#---------------------------------------------------------------
#   target definitions
#---------------------------------------------------------------

#object file
DynAlg.o: DynAlg.c DynAlg.h Movements.h Movements.c
cc $(CFLAGS) $<

#executable file
DynAlg: DynAlg.o
$(LD) $(LDFLAGS) -o DynAlg \
$(ASDP_LIB)/start.o \
DynAlg.o \
$(LLIBS) \
        $(LLIBS2)

# S record file
DynAlg.mx: DynAlg
$(SFORMAT) -o DynAlg.mx DynAlg

#---------------------------------------------------------------
#   load an executable file
#---------------------------------------------------------------

load: DynAlg.mx
$(SLOAD) -d /dev/ttya DynAlg.mx
```

C.3 File DynAlg.h

```
/*..............................................................
DynAlg.h - header file containing the global variables for
DynAlg.c, an Integrated Architecture for Learning, Planning and
Reacting.
................................................................
Unit:
File:
Date:
Author:
Version:
Revisions:
Comments:

..............................................................*/
#define XMAX 8   /* number of WorldModelCells in x direction */
#define YMAX 5   /* number of WorldModelCells in y direction */
#define TRIALMAX 100    /* maximum number of trials         */
#define DX 1            /* magnitude of step in x directiono */
#define DY 1            /* magnitude of step in y directiono */
#define ACTIONS 4       /* number of different actions that can
                                 be taken from a state (cell)*/
#define STATEDIM 3      /* dimension of the state vector     */
#define GAMMA 900  /* /1000 used to calculate posterior eval */
#define ALFA 1000  /* used to update the policy table        */
#define BETA 100   /* /1000 used to update the evaluatn table*/

#define EVALU  91  /* constant for the PrintResults routine  */
#define POLICY 92  /* constant for the PrintResults routine  */

typedef int *pstate;    /* pointer to a state in world model */
typedef int state;      /* states in the world model         */
enum statecomp { ANG, X, Y };

typedef int boolean;    /* define boolean type               */
enum booleanset { FALSE, TRUE };

typedef int switcher;   /* define switcher type              */
enum switcherset { OFF, ON };

typedef int action;     /* action to be taken from a state   */
enum actionset { UP, DOWN, LEFT, RIGHT };

typedef int infotype;   /* informations for a cell in model  */
enum infoset { UNKNOWN, EMPTY, OCCUPIED };

struct worldcell {
  infotype Info; /*contains info about the cell as in infoset*/
  boolean ActionKnown[ACTIONS];
  int Reward;               /* reward for reaching the cell */
  int Frequency; /* contains the number of times the cell ...*/
} WorldModelCell[XMAX][YMAX];        /* ... has been reached */

struct mapinfo{
  int Frequency[XMAX][YMAX]; /* saves the Frequency from
                                   worldmodel at each  trial */
  int RobotMotions;  /* saves the MoveCounter for each trial */
  int DoRealActions; /* saves the DoActionCounter for each
                                                       trial */
} Map[TRIALMAX];

int MoveCounter = 0;    /* counts the movements of the robot */
int DoActionCounter = 0; /* counts the number of times the
                             routine DoRealAction is invoked */

double Eval[XMAX][YMAX];       /* the evaluation table       */

double Policy[XMAX][YMAX][ACTIONS];  /* the policy table     */

state RobotPos[STATEDIM];  /* position of the robot for sonar*/
boolean SonarAlarm=FALSE;  /* obstacle avoidance             */
boolean GoalFound=FALSE;   /* set by infrared sensor         */
boolean GoalPointSet=FALSE;  /*goal has not been located yet */
int XGoal, YGoal;            /* position of the goal state   */

switcher Simul=OFF;          /* simulation mode: on or off   */
switcher SimObst=ON;         /* simulation of obstacles      */
```

C.4 File DynAlg.c

```
/*..............................................................
  DynAlg.c - file containing the algorithm for Dyna,
             an Integrated Architecture for Learning, Planning
             and Reacting.
...............................................................
Unit:
File:  DynAlg.c
Date:  16 Jan 1995
Author:  Alex Weiser
Version:  0.1
Revisions:
Comments:




Functions:
  RealStep(RealState)
  HypotStep(HypotState)
  Evaluate(State)
  ChooseAction(CurrState,pAction)
  GetNextState(State,Action,NextState)
  DoRealAction(RealState,Action,NextState)
  FormPostEval(NewState,ActionReward)
  UpdateWorldModel(State,NextState,Info,ActionReward)
  UpdatePolicyEval(State,PriorEval,PostEval,Action)
  GetReward(State)
  IsGoal(State)
  Initialise()
  SetParameters(pHypotCycles,pTrialMax,StartState)
  MakeTables(Table)
  DrawMap(Trial)
  PrintResults(Trial)
  ShowMap(Trial)
.............................................................*/

#include "libcalbh.h"
#include "libcalbl.h"
#include "libcpro.h"
#include "libcstd.h"
#include "libio.h"
#include "DynAlg.h"
#include "Movements.h"
#include "Movements.c"

/*..............................................................
RealStep(RealState)- routine to execute a step on the real
world.
.............................................................*/
void RealStep(RealState)
pstate RealState;
{
int i, ActionReward;
double PriorEval,PostEval;
action Action;
infotype StateInfo;
state NextState[STATEDIM], OldRealState[STATEDIM];
infotype GetNextState();
double Evaluate(), FormPostEval();
boolean DoRealAction(), ChooseAction();
void UpdateWorldModel(), UpdatePolicyEval();
extern struct worldcell WorldModelCell[XMAX][YMAX];

  PriorEval = Evaluate(RealState);

  ChooseAction(RealState,&Action);

  for(i=0;i<STATEDIM;i++) OldRealState[i] = RealState[i];

  StateInfo = GetNextState(RealState,Action,NextState);

  if (StateInfo!=OCCUPIED && DoRealAction(RealState,Action,
    NextState)){
    StateInfo = EMPTY;
    ActionReward = GetReward(RealState);
    PostEval = FormPostEval(NextState,ActionReward);
  }
  else{
    StateInfo = OCCUPIED;
    ActionReward = 0;
    PostEval = PriorEval;
  }

  UpdateWorldModel(OldRealState,NextState,StateInfo,
    ActionReward);
  UpdatePolicyEval(OldRealState,PriorEval,PostEval,Action);

} /* RealStep */

/*..............................................................
HypotStep(HypotState)- routine to execute a hypothetical step
on the world model.
..............................................................*/
void HypotStep(HypotState)
pstate HypotState;
{
int ActionReward;
double PriorEval, PostEval;
action Action;
state NextState[STATEDIM];
int GetNextState();
boolean ChooseAction();
double Evaluate(), FormPostEval();
void UpdatePolicyEval();

  PriorEval = Evaluate(HypotState);

  do{
    ChooseAction(HypotState,&Action);
  }while(GetNextState(HypotState,Action,NextState) != EMPTY);

  ActionReward = WorldModelCell[NextState[X]][NextState[Y]].
    Reward;
  PostEval = FormPostEval(NextState,ActionReward);

  UpdatePolicyEval(HypotState,PriorEval,PostEval,Action);

} /* HypotStep */

/*..............................................................
Evaluate(State)- returns the heuristic reward written in the
```

```
evaluation table for the specified state.
..............................................................*/
double Evaluate(State)
pstate State;
{
extern double Eval[XMAX][YMAX];

  return(Eval[State[X]][State[Y]]);

} /* Evaluate */

/*..............................................................
ChooseAction(State,pAction)- decides the action to be taken
from the given state by checking the policy table. In the case
that the policy table is empty for that state, the action is
chose randomly.
..............................................................*/
boolean ChooseAction(State,pAction)
pstate State;
action *pAction;
{
int i;
double RandNumb, ProbabSum, drand48();
double UpToDown, DownToLeft, LeftToRight;
extern double Policy[XMAX][YMAX][ACTIONS];
extern struct worldcell WorldModelCell[XMAX][YMAX];

  ProbabSum = 0.0;

  for(i=0;i<ACTIONS;i++)
    ProbabSum += exp(Policy[State[X]][State[Y]][i]);

  UpToDown = exp(Policy[State[X]][State[Y]][UP])/ProbabSum;
  DownToLeft = exp(Policy[State[X]][State[Y]][DOWN])/ProbabSum
    + UpToDown;
  LeftToRight = exp(Policy[State[X]][State[Y]][LEFT])/ProbabSum
    + DownToLeft;

  while(TRUE){
    RandNumb = drand48();
    if(RandNumb>0 && RandNumb<UpToDown)
      *pAction=UP;
    else if (RandNumb>UpToDown && RandNumb<DownToLeft)
      *pAction=DOWN;
    else if (RandNumb>DownToLeft && RandNumb<LeftToRight)
      *pAction=LEFT;
    else if (RandNumb>LeftToRight && RandNumb<1)
      *pAction=RIGHT;
    else
      continue;
    break;
  }

} /* Evaluate */

/*..............................................................
GetNextState(State,Action,NewState)- plans the following step
calculating the new position. Returns an integer value
according to the information about the new State.
..............................................................*/
infotype GetNextState(State,Action,NewState)
```

```
action Action;
pstate NewState, State;
{
int pos[STATEDIM];
extern struct worldcell WorldModelCell[XMAX][YMAX];

  switch(Action){
    case UP:
      pos[X] = State[X];
      pos[Y] = State[Y] - DY;
      break;
    case DOWN:
      pos[X] = State[X];
      pos[Y] = State[Y] + DY;
      break;
    case LEFT:
      pos[X] = State[X] - DX;
      pos[Y] = State[Y];
      break;
    case RIGHT:
      pos[X] = State[X] + DX;
      pos[Y] = State[Y];
      break;
  }

  NewState[X] = pos[X];
  NewState[Y] = pos[Y];
  NewState[ANG] = State[ANG];

  return(WorldModelCell[NewState[X]][NewState[Y]].Info);

} /* GetNextState */

/*..............................................................
DoRealAction(RealState,Action,NextState)- executes the real
action with the robot and returns a boolean value according
to the success of the action, i.e., presence or not of an
obstacle.
.............................................................*/
boolean DoRealAction(RealState,Action,NextState)
action Action;
pstate RealState, NextState;
{
int Angle, AngleDiff, Axsis;
extern int DoActionCounter;
boolean Rotate(), Transalte();
void GenerateSonarAlarm();

  DoActionCounter++;

  switch(Action){
    case UP:
      Angle = 90;  Axsis = -Y; break;
    case DOWN:
      Angle = 270; Axsis = Y;  break;
    case LEFT:
      Angle = 180; Axsis = -X; break;
    case RIGHT:
      Angle = 0;   Axsis = X;  break;
  }
  AngleDiff = Angle-RealState[ANG];

  if(SimObst)
    GenerateSonarAlarm(RealState,Action);

  if(Simul){
    GenerateSonarAlarm(RealState,Action);
    if(!SonarAlarm){
      RealState[X] = NextState[X];
      RealState[Y] = NextState[Y];
      RealState[ANG] = Angle;
      return(TRUE);
    }
    else{
      SonarAlarm = FALSE;
      return(FALSE);
    }
  }
  else{
    if(Rotate(AngleDiff))
      RealState[ANG] = Angle;
    else
      return(FALSE);
    if(Translate(Axsis)){
      RealState[X] = NextState[X];
      RealState[Y] = NextState[Y];
      return(TRUE);
    }
    else
      return(FALSE);
  }

} /* DoRealAction */

/*.........................................................
FormPostEval(NewState,Reward)- recalculates the evaluation of
the old state. Value used for eventual corrections in the
policy.
.......................................................*/
double FormPostEval(NewState,Reward)
pstate NewState;
int Reward;
{
double PostEval;
double Evaluate();

  PostEval = (double)Reward + (double)GAMMA/1000.0 *
    Evaluate(NewState);

  return(PostEval);

} /* FormPostEval */

/*.........................................................
UpdateWorldModel(State,NextState,Info,ActionReward)- writes
the reward, learned from the real world, in the world model.
.......................................................*/
void UpdateWorldModel(State,NextState,Info,ActionReward)
pstate State,NextState;
infotype Info;
int ActionReward;
{
```

```
extern struct worldcell WorldModelCell[XMAX][YMAX];

  WorldModelCell[NextState[X]][NextState[Y]].Info = Info;
  WorldModelCell[NextState[X]][NextState[Y]].Reward =
    ActionReward;

  if (Info==EMPTY)
    WorldModelCell[NextState[X]][NextState[Y]].Frequency++;
  else
    WorldModelCell[State[X]][State[Y]].Frequency++;

} /* UpdateWorldModel */

/*...............................................................
UpdatePolicyEval(State,PriorEval,PostEval,Action)- updates the
policy and the evaluation table.
..............................................................*/
void UpdatePolicyEval(State,PriorEval,PostEval,Action)
pstate State;
double PriorEval, PostEval;
action Action;
{
double EvalDiff, PolicyIncr;
extern double Eval[XMAX][YMAX];
extern double Policy[XMAX][YMAX][ACTIONS];

  EvalDiff = PostEval - PriorEval;
  Eval[State[X]][State[Y]] += (double)BETA/1000.0 * EvalDiff;
  PolicyIncr= (double)ALFA/1000.0 * EvalDiff;
  Policy[State[X]][State[Y]][Action] += PolicyIncr;

} /* UpdatePolicyEval */

/*...............................................................
GetReward(State)- returns the reward of the new state.
The reward should be supplied by a reaction from the real world
..............................................................*/
int GetReward(State)
state State[STATEDIM];
{
extern boolean GoalPointSet;

  if (GoalPointSet && State[X]==XGoal && State[Y]==YGoal)
    return(1);
  else
    return(0);

} /* GetReward */

/*...............................................................
IsGoal(State)- returns TRUE when the state corresponds to the
goal. First it checks the signal 'GoalFound' and then the last
values in XGoal and YGoal.
..............................................................*/
boolean IsGoal(State)
pstate State;
{
extern int XGoal, YGoal;
extern boolean GoalFound, GoalPointSet;

  if (GoalFound){    /* boolean value set by infrared sensor */
    XGoal = State[X];
    YGoal = State[Y];
    GoalFound = FALSE;
    GoalPointSet = TRUE;
    return(TRUE);
  }
  else if (GoalPointSet && State[X]==XGoal && State[Y]==YGoal)
    return(TRUE);
  else
    return(FALSE);

} /* IsGoal */

/*.........................................................
Initialise()- initialises the evaluation table, the policy
table and the world model.
.........................................................*/
void Initialise()
{
int i, j, k;
long int time1,timer();
extern double Eval[XMAX][YMAX];
extern double Policy[XMAX][YMAX][ACTIONS];
extern struct worldcell WorldModelCell[XMAX][YMAX];

  printf("\nInitialising...\n");

  for(i=0;i<XMAX;i++){
    for(j=0;j<YMAX;j++){
      Eval[i][j]=0.0;
      WorldModelCell[i][j].Info=UNKNOWN;
      WorldModelCell[i][j].Reward=0;
      WorldModelCell[i][j].Frequency=0;
      for(k=0;k<ACTIONS;k++){
        WorldModelCell[i][j].ActionKnown[k]=FALSE;
        Policy[i][j][k]=0.0;
      }
    }
  }

} /* Initialise */

/*.........................................................
SetParameters(pHypotCycles,pTrialMax,StartState)- this routine
prompts the user to enter parameters.
.........................................................*/
void SetParameters(pHypotCycles,pTrialMax,StartState)
int *pHypotCycles, *pTrialMax;
pstate StartState;
{
int Seed;
extern int XGoal, YGoal;
void srand48();

  printf("\nEnter number of hypothetical cycles: ");
  scanf("%d",pHypotCycles);

  do{
    printf("\nEnter number of trials: ");
    scanf("%d",pTrialMax);
  }while(*pTrialMax>TRIALMAX);
  printf("\nEnter seed (unsigned short): ");
  scanf("%d",&Seed);
  srand48(Seed);

  printf("\nEnter start state (x, y, ang): ");
  StartState[X]=1;
  StartState[Y]=1;
  StartState[ANG]=270;
  WorldModelCell[StartState[X]][StartState[Y]].Info = EMPTY;

  printf("(%d, %d, %d)\n",StartState[X],StartState[Y],
    StartState[ANG]);

  printf("\nAssign goal state (x,y): ");
  scanf("%d  %d",&XGoal,&YGoal);
  GoalPointSet = TRUE;
  printf("  (%d,%d)\n",XGoal,YGoal);

  /*This should be printed in the Output file from the
    routine PrintResults

    printf("\nNumber of hypothetical cycles: %d",HypotCycles);
      printf("\nNumber of trials: %d",TrialMax);
      printf("\nSeed (unsigned short): %d",Seed);
      printf("\nStart state: ");
      printf("(%d, %d, %d)\n",StartState[X],StartState[Y],
        StartState[ANG]);
      printf("\nGoal state: %d  %d",XGoal,YGoal);
  */

} /* SetParameters */

/*............................................................
MakeTables(Table)- writes the values of the evaluation table
and the policy table in the standard output.
............................................................*/
void MakeTables(Table)
int Table;
{
int i,j, k, PolicyXBegin;
double ProbabSum[XMAX];
extern double Eval[XMAX][YMAX];
extern double Policy[XMAX][YMAX][ACTIONS];
extern struct worldcell WorldModelCell[XMAX][YMAX];

  switch(Table){
    case EVALU:                    /* print Evaluation Table */
      printf("\nEvaluation Table\n\n");
      for(k=0;k<YMAX;k++){
        for(i=0;i<XMAX;i++){
          printf("  %6.4f ",Eval[i][k]);
        }
        printf("\n\n");
      }
      break;

    case POLICY:                       /* print Policy Table */
      PolicyXBegin=0;
      printf("\nPolicy Table\n\n", PolicyXBegin);
      for(i=PolicyXBegin;i<XMAX;i++){
        printf("+ -    - ");
      }
      printf("+\n");

      for(k=0;k<YMAX;k++){
        for(i=PolicyXBegin;i<XMAX;i++){
          ProbabSum[i]=0.0;
          for(j=0;j<ACTIONS;j++)
            ProbabSum[i] += exp(Policy[i][k][j]);
        }

        for(i=PolicyXBegin;i<XMAX;i++){
          printf("|  %4.0f  ",exp(Policy[i][k][UP])/
            ProbabSum[i]*100.);
        }
        printf("|\n");

        for(i=PolicyXBegin;i<XMAX;i++){
          printf(" %4.0f%4.0f",
            exp(Policy[i][k][LEFT])/ ProbabSum[i]*100.,
            exp(Policy[i][k][RIGHT])/ProbabSum[i]*100.);
        }
        printf("\n");

        for(i=PolicyXBegin;i<XMAX;i++){
          printf("|  %4.0f  ",
            exp(Policy[i][k][DOWN])/ProbabSum[i]*100.);
        }
        printf("|\n");
        for(i=PolicyXBegin;i<XMAX;i++){
          printf("+ -    - ");
        }
        printf("+\n");

      }
      printf("\n");
      break;

    default:
      break;

  }

} /* MakeTables */


/*...............................................................
DrawMap(Trial)- draws a map with the path followed by the robot
in the last real experience.
...............................................................*/
void DrawMap(Trial)
int Trial;
{
int i, k;
extern int MoveCounter, DoActionCounter;
extern struct worldcell WorldModelCell[XMAX][YMAX];
extern struct mapinfo Map[TRIALMAX];

  for(k=0;k<YMAX;k++){
    for(i=0;i<XMAX;i++){
      if(WorldModelCell[i][k].Info==EMPTY)
        Map[Trial].Frequency[i][k] = WorldModelCell[i][k].
          Frequency;
      else if(WorldModelCell[i][k].Info==OCCUPIED)
        Map[Trial].Frequency[i][k] = -1;
      else
        Map[Trial].Frequency[i][k] = 0;
      WorldModelCell[i][k].Frequency=0;   /* reset the value */
    }
  }

   Map[Trial].RobotMotions = MoveCounter;
   Map[Trial].DoRealActions = DoActionCounter;
   MoveCounter = 0;
   DoActionCounter = 0;

} /* DrawMap */

/*.........................................................
PrintResults(Trial)- prints the results on the standard output.
........................................................*/
void PrintResults(PathFound,Trial,RealIter)
boolean PathFound;
int Trial, RealIter;
{
int SeeTrial,i;
char *TextLine;
long int time1,timer();
void MakeTables(), ShowMap();
void GetText(),Report();

  if(PathFound){
    printf("\nOptimal path found in %d trials.\n",Trial-1);
    printf("Need %d steps to reach goal.\n\n",RealIter);
  }
  else
    printf("\nCouldnt find optimal path in %d trials.\n\n",
      Trial-1);

  do{
    printf("\nWhich trial do you want to see ");
    printf("(1-%d) ? ",Trial);
    printf("\n91: evaluation table");
    printf("\n92: policy table");
    printf("\n99: write everything in a file");
    printf("\n0:  quit\n");
    scanf("%d",&SeeTrial);
    printf("\n");

    if(SeeTrial==0) break;
    MakeTables(SeeTrial);
    if(Simul && SeeTrial==99){
      printf("\nSorry, not implemented in simulation mode.\n");
      continue;
    }
    if(SeeTrial==99){
      Report("\nPlease finish tip connection, ");
      Report("make a redirection\n\r");
      Report("to a file with: cat /dev/ttya >> ");
      Report("<filename>\n\r");
      Report("and press <Enter> ...\n\r");
      GetText(TextLine);
      Report("\n\r");
      /* print information typed in the SetParamaters routine*/
      if(PathFound){
        printf("\nOptimal path found in %d trials.\n",Trial-1);
        printf("Need %d steps to reach goal.\n\n",RealIter);
      }
      else
        printf("\nCouldnt find optimal path in %d trials.\n\n",
          Trial-1);
      for(i=1;i<=Trial;i++)
        ShowMap(i);
      MakeTables(EVALU);
      MakeTables(POLICY);
      Report("\nPlease cancel redirection, ");
      Report("restore tip connection\n\r");
      Report("and press <Enter> ...\n\r");
      GetText(TextLine);
      Report("\n\r");
    }
    if(SeeTrial>Trial) continue;
    ShowMap(SeeTrial);

  }while(TRUE);

  time1=timer();
  while(timer()<(time1+(long)100));         /* wait 1 second */
  close_port2();


} /* PrintResults */

/*.............................................................
ShowMap(Trial)- draws a map with the path followed by the robot
in the real experience.
.............................................................*/
void ShowMap(Trial)
int Trial;
{
int i, k, sum = 0;
extern struct mapinfo Map[TRIALMAX];

  printf("\n\nREAL EXPERIENCE %d \n\n",Trial);
  for(k=0;k<YMAX;k++){
    for(i=0;i<XMAX;i++){
      if(Map[Trial].Frequency[i][k] == -1)
        printf("# \t");
      else if(Map[Trial].Frequency[i][k] == 0)
        printf(" \t");
      else{
        printf("%d \t",Map[Trial].Frequency[i][k]);
        sum+=Map[Trial].Frequency[i][k];
      }
    }
    printf("\n\n");
  }
  printf(" %d steps %d calls %d motions\n\n",sum,
    Map[Trial].DoRealActions,Map[Trial].RobotMotions);
```

```
} /* ShowMap */

Procs_To_Add tab1 = {1,{MotionProcess,10,1,0}};
Procs_To_Remove tab2 = {1,MotionProcess};

/*...........................................................
main() - main routine with the loops for the real and
hypothetical steps.
...........................................................*/
main()
{
char c;
int HypotCycles, Trial, TrialMax, i, k, RealIter, LastRealIter;
int MinIter;
state HypotState[STATEDIM], StartState[STATEDIM];
boolean PathFound;
extern struct worldcell WorldModelCell[XMAX][YMAX];
extern state RobotPos[STATEDIM];
extern int XGoal, YGoal;
boolean IsGoal();
double drand48();
void Initialise(), SetParameters(), DrawMap(), PrintResults();
void InitMovements(),TermMovements(), ReturnRobot();

  Initialise();
  SetParameters(&HypotCycles,&TrialMax,StartState);
  if(!Simul) InitMovements();

  printf("ALFA = %d\n",ALFA);
  printf("BETA = %d/1000\n",BETA);
  printf("\nWorking...\n");
  Trial = 0;
  MinIter = 10000;
  LastRealIter = 0;
  do{
    Trial++;
    for(i=0;i<STATEDIM;i++)  RobotPos[i]=StartState[i];

    RealIter = 0;
    do{
      RealIter++;
      RealStep(RobotPos);                   /* do a real step */
      if(!Simul && (RealIter>10) && (RealIter&8) &&
         !(RealIter&4) && !(RealIter&2) && !(RealIter&1))
         ReturnRobot();

      if(Trial>1){
        /* start a cycle of hypothetical trials */
        for(k=0;k<HypotCycles;k++){
          do{
            HypotState[X]=(int)(drand48()*(double)XMAX);
            HypotState[Y]=(int)(drand48()*(double)YMAX);
          }while(WorldModelCell[HypotState[X]][HypotState[Y]].
          Info!=EMPTY || IsGoal(HypotState));
          HypotStep(HypotState);   /* do a hypothetical step */
        }
      }
    }while(!IsGoal(RobotPos));
    printf("\n%d RealIter\n",RealIter);
    DrawMap(Trial);
    /* check for optimal path and maximum number of trials */
    if(RealIter==LastRealIter && RealIter==MinIter){
      PathFound = TRUE;
      break;
    }
    else if (Trial>=TrialMax){
      PathFound = FALSE;
      break;
    }
    else if (!Simul) ReturnRobot();   /* bring robot to start
                                                    position */

    if(RealIter<=MinIter)
      MinIter = RealIter;
    LastRealIter = RealIter;

  }while(TRUE);

  if (!Simul) TermMovements();
  PrintResults(PathFound,Trial,RealIter);

} /* main */
```

C.5 File Movements.h

```
/*..............................................................
Movements.h - header file containing the global variables for
Movements.c
..............................................................
Unit:
File:
Date:
Author:
Version:
Revisions:
Comments:

.............................................................*/

#define ROTAT_VEL 20          /* deg/s : rotation velocity   */
#define TRANS_VEL 200         /* mm/s : translation velocity */
#define DISTANCE 2000         /* mm : length of one motion step
                                               of the robot */
#define MIN_DIST1 3500        /* minimum distance from obstacle
                                               before move  */
#define DIST_OS   2000        /* add to MIN_DIST1 in LEFTS,
                                          RIFGHTS and BACKS */
#define MIN_DIST2 1200        /* minimum distance from obstac
                                             at end of move */
#define MIN_DELAY 5L          /* x10ms for TSVME-120. Delay
                                           between us firing*/
short int DevNumb = 1;
long int MotnStatus[2], Position;
char *Error;             /* adress of Albatros error messages*/
boolean DebugOn=TRUE;    /* debug parameter, if ON status
                            messages are printed            */
int SonarOn;
boolean Obstacle = FALSE;      /* flag indicating the presence
                        of an obstacle located by the sonars */

int Offset[STATEDIM];          /* offset of robot position   */

struct robotdata {
  short int Communic;
  short int Code;
  short int Status;
  short int MotionType;
  char *Direction;
  int Magnitude, Velocity;
} Robot;

enum CommSet   { CLEAR, NEW_COMMAND, ACK, EM_STOP,INITIALISE };
enum CodeSet   { EXECUTING, READY ,ABORTED };
enum StatusSet { INIT, OK, OBSTACLE, TROUBLE };
enum MotionTypeSet   { ROTATION, TRANSLATION };
enum Sensors   { LEFTS, MIDDLES, RIGHTS };
enum SideSet   { FRONTSD, BACKSD, RIGHTSD, LEFTSD };
```

C.6 File Movements.c

```
/*..............................................................
Movements.c - contains the routines to move the platform. The
routines are used by the DynAlg.
...............................................................
Unit:
File:
Date:
Author:
Version:
Revisions:
Comments:
...............................................................
Functions:                                                   */

long    timer();
boolean Rotate(),
Translate(),
        Scan();
void MotionProcess(),
        ReadFront(),
        ReadBack(),
        ReadRight(),
        ReadLeft(),
InitMovements(),
        ReturnRobot(),
TermMovements(),
GenerateSonarAlarm(),
        PrintStatus(),
        Report(),
        GetText();

/*..............................................................
timer() - used for delaysbetween sensor readings.
Notes:      --> TSVME-120 - Basic clock cycle = 10 ms
.............................................................*/
long int timer()
{
static long *time=(long *)NULL;

  if (time==NULL) sendf("TSYS C=5 O=%lx",&time);
  return *time;

} /* timer */

/*..............................................................
Rotate(Angle)- rotates the robot to the propper
 orientation in order to make the newa translation.
.............................................................*/
boolean Rotate(Angle)
int Angle;
{
int i,Side;
extern struct robotdata Robot;
void Report();

  /* avoid rotations with more than 180 degrees */
  if (Angle>180)
    Angle+=-360;
  else if (Angle<-180)
```

```
    Angle+=360;

  switch(Angle){
    case    0: Side = FRONTSD; Report("Scan front\r\n"); break;
    case  -90: Side = RIGHTSD; Report("Scan right\r\n"); break;
    case   90: Side = LEFTSD;  Report("Scan left\r\n");  break;
    case  180:
    case -180: Side = BACKSD;  Report("Scan back\r\n");  break;
  }

  Scan(Side,CLEAR);

  if(SonarAlarm){
    Obstacle = TRUE;
    SonarAlarm = FALSE;
  }

  for(i=0;i<6;i++)
    while(!Scan(Side,MIN_DIST1));   /* make 6 sonar readings */

  if (Angle!=0){
    Robot.MotionType = ROTATION;
    if(Angle>0)
      Robot.Direction = "LEFT";
    else
      Robot.Direction = "RIGHT";
    Robot.Magnitude = abs(Angle) - Offset[ANG];
    Robot.Velocity = ROTAT_VEL;
    Robot.Communic = NEW_COMMAND;

    while(Robot.Communic != ACK);
    while(Robot.Code != READY){
      if(Robot.Status == OBSTACLE){
        Robot.Status = OK;
        Robot.Code = READY;
        return(FALSE);
      }
    }
    Offset[ANG] = Angle / abs(Angle) *
                  ((int)Position-Robot.Magnitude);
  }
  if(DebugOn) printf("Offset[ANG]=%d\n",Offset[ANG]);
  return(TRUE);

} /* Rotate */

/*..............................................................
Translate(Achsis)- executes the movement of the robot one step
forward. Returns a boolean value corresponding to the
possibilty to execute the action, e.g., no obstacle.
..............................................................*/
boolean Translate(Achsis)
int Achsis;
{
int i;
boolean PositionReached = TRUE;
extern struct robotdata Robot;
extern long int MotnStatus[2];
extern boolean SonarOn;
long int send(), sendf();
```

```
  Robot.MotionType = TRANSLATION;
  Robot.Direction = "FORWARD";
  SonarOn = TRUE;
  Robot.Magnitude = DISTANCE - Offset[abs(Achsis)];
  Robot.Velocity = TRANS_VEL;
  Robot.Communic = NEW_COMMAND;
  while(Robot.Communic != ACK);
  while(Robot.Code != READY){
    switch (Robot.Status){
      case OBSTACLE:
        if(Robot.Code != ABORTED){
          Robot.Direction = "KILL";
          Robot.Communic = EM_STOP;
          while(Robot.Communic != ACK);
          while(Robot.Code != READY);
          Robot.Direction = "BACKWARD";
          Robot.Magnitude = (int)Position+Offset[abs(Achsis)];
          Robot.Communic = NEW_COMMAND;
          Achsis = -Achsis;
          while(Robot.Communic != ACK);
          while(Robot.Code != READY);
        }
        else
          Robot.Code = READY;

        Robot.Status = OK;
        PositionReached = FALSE;
        break;
      case OK:
        break;
    }
  }
  SonarOn = FALSE;
  Offset[abs(Achsis)]=Achsis/abs(Achsis)*((int)Position-Robot.
    Magnitude);
  if(DebugOn) printf("Offset[%d]=%d\n",abs(Achsis),
    Offset[abs(Achsis)]);
  return(PositionReached);

} /* Translate */

/*.........................................................
MotionProcess() - routine running as an interruptible process.
.........................................................*/
void MotionProcess()
{
float RelPosError;
long int Moving, Error, InitPos, Threshold;
extern int MoveCounter;
extern boolean Obstacle, SonarOn;
extern long int MotnStatus[2];
extern struct robotdata Robot;
long int sendf(), send();
int strcmp();

  if (Robot.Communic==EM_STOP){
    if(DebugOn) PrintStatus();
    send("MOTN KILL\r");
    Robot.Code=EXECUTING;
    Robot.Communic=ACK;
    if(DebugOn) PrintStatus();
    return;
```

```
}

if (Robot.Communic==NEW_COMMAND){            /* new command */
  if(DebugOn) PrintStatus();
  if(Obstacle){
    Robot.Code = ABORTED;
    Obstacle = FALSE;
    Robot.Communic = ACK;
    return;
  }
  switch(Robot.MotionType){
    case ROTATION:
      Error=sendf("MOTN %s A=%d V=%d",Robot.Direction,
        Robot.Magnitude, Robot.Velocity);
      break;
    case TRANSLATION:
      MoveCounter++;
      Error=sendf("MOTN %s D=%d V=%d",Robot.Direction,
        Robot.Magnitude, Robot.Velocity);
      break;
  }
  if(Error==0)
    Robot.Code = EXECUTING;
  else {
    if(DebugOn) printf("***\t Error %d\n",Error);
    Robot.Code = ABORTED;
  }
  Robot.Communic = ACK;
  if(DebugOn) PrintStatus();
  /*sendf("MOTN STATUS 0=%x\n",MotnStatus);
  InitPos = MotnStatus[1];*/
  return;
}

if(Robot.Communic==CLEAR){
  if(DebugOn) PrintStatus();
  /* clear encoders */
  Robot.Code = READY;
  Robot.Communic = ACK;
  if(DebugOn) PrintStatus();
  return;
}

if(Robot.Communic==INITIALISE){
  Robot.Code = READY;          /* initialisation completed */
  Robot.Communic = ACK;
  if(DebugOn) PrintStatus();
  return;
}

/* update Robot.Code by monitoring the status */

if (Robot.Code==ABORTED){
  Robot.Magnitude = 0;
  Position = 0;
  Robot.Status = OBSTACLE;
  if(DebugOn) PrintStatus();
  return;
}

if (Robot.Code==EXECUTING){
  sendf("MOTN STATUS 0=%x\n",MotnStatus);
```

```
    Moving=MotnStatus[0];
    InitPos = (long)0;
    Position=MotnStatus[1]-InitPos;
    if(Moving && SonarOn){
      Threshold = MIN_DIST1 - (MIN_DIST1 - MIN_DIST2) *
        (int)Position/Robot.Magnitude;
      Scan(FRONTSD,Threshold);
      if(Obstacle){
        Robot.Status = OBSTACLE;        /* obstacle detected */
        SonarOn = FALSE;
        Obstacle = FALSE;
        if(DebugOn) PrintStatus();
        if(DebugOn) printf("Encoder %ld\n",Position);
      }
    }
    else if(!Moving) {
      Robot.Direction = "STOPPED";
      Robot.Code = READY;
      RelPosError=(float)(Position-Robot.Magnitude)/
        (float)Robot.Magnitude;

      if(RelPosError<0.1 && RelPosError>-0.1)
        Robot.Status=OK;            /* final postion reached */
      else if(RelPosError==-1.0)
        printf("*** Emergency Stop is pressed!\n");
      else
        Robot.Status = TROUBLE;

      if(DebugOn) PrintStatus();
      if(DebugOn) printf("Encoder %ld\n",Position);
    }
    return;
  }

} /* MotionProcess */

/*.........................................................
Scan(Side,Threshold) - makes a scan in the direction of motion
an decides if there isan obstacle or not by setting the global
variable SonarAlarm.
.......................................................*/
boolean Scan(Side,Threshold)
int Side,Threshold;
{
int i;
static long int Distance[3]={'0','0','0'};
static long int time1 = -MIN_DELAY;
boolean Scanned = FALSE;
extern boolean Obstacle;
long int timer();
void Report();

  if(Threshold==CLEAR){
    for(i=0;i<3;i++) Distance[i]=0;
    Obstacle = FALSE;
    return;
  }

  if((timer()-time1)>MIN_DELAY){
    switch(Side){
      case FRONTSD:  ReadFront(Distance);
```

```
                     break;
       case BACKSD:  ReadBack(Distance);
                     Threshold += DIST_OS;
      break;
       case RIGHTSD: ReadRight(Distance);
                     Threshold += DIST_OS;
                     break;
       case LEFTSD:  ReadLeft(Distance);
                     Threshold += DIST_OS;
      break;
     }
     time1=timer();
     Scanned = TRUE;
  }

  /* Set Obstacle according to SensorValue */
  if((Distance[0]>0)&&(Distance[1]>0)&&(Distance[2]>0)){
    if(((Distance[0]<Threshold) + (Distance[1]<Threshold) +
      (Distance[2]<Threshold)) >= 2){
      Distance[0]=0; Distance[1]=0; Distance[2]=0;
      Obstacle = TRUE;
      Report("Sonar Alarm\n\r");
    }
  }
  return(Scanned);

} /* Scan */

/*...............................................................
ReadFront(pDistance) - reads alternately all three front sensor
values.
..............................................................*/
void ReadFront(pDistance)
long int *pDistance;
{
static NextSensor = LEFTS;
long int SensorValue[20];

  switch(NextSensor){
    case LEFTS:
      sendf("READ C=2,001000,000000 O=%x",SensorValue);
      pDistance[LEFTS]=*(SensorValue+2);
      NextSensor = MIDDLES;
      printf("L %ld\t",pDistance[LEFTS]);
      break;
    case MIDDLES:
      sendf("READ C=1,100000,000000 O=%x",SensorValue);
      pDistance[MIDDLES]=*SensorValue;
      NextSensor = RIGHTS;
      printf("C %ld\t",pDistance[MIDDLES]);
      break;
    case RIGHTS:
      sendf("READ C=5,001000,000000 O=%x",SensorValue);
      pDistance[RIGHTS]=*(SensorValue+2);
      NextSensor = LEFTS;
      printf("R %ld\t",pDistance[RIGHTS]);
      break;
  }

} /* ReadFront */

/*...............................................................
```

```
ReadBack(pDistance) - reads alternately all three front sensor
values.
.............................................................*/
void ReadBack(pDistance)
long int *pDistance;
{
static NextSensor = LEFTS;
long int SensorValue[20];

  switch(NextSensor){
    case LEFTS:
      sendf("READ C=1,010000,000000 O=%x",SensorValue);
      pDistance[LEFTS]=*(SensorValue+1);
      NextSensor = MIDDLES;
      printf("L %ld\t",pDistance[LEFTS]);
      break;
    case MIDDLES:
      sendf("READ C=4,100000,000000 O=%x",SensorValue);
      pDistance[MIDDLES]=*SensorValue;
      NextSensor = RIGHTS;
      printf("C %ld\t",pDistance[MIDDLES]);
      break;
    case RIGHTS:
      sendf("READ C=2,010000,000000 O=%x",SensorValue);
      pDistance[RIGHTS]=*(SensorValue+1);
      NextSensor = LEFTS;
      printf("R %ld\t",pDistance[RIGHTS]);
      break;
  }

} /* ReadBack */

/*.............................................................
ReadRight(pDistance) - reads alternately all three front sensor
values.
.............................................................*/
void ReadRight(pDistance)
long int *pDistance;
{
static NextSensor = LEFTS;
long int SensorValue[20];

  switch(NextSensor){
    case LEFTS:
      sendf("READ C=2,100000,000000 O=%x",SensorValue);
      pDistance[LEFTS]=*SensorValue;
      NextSensor = MIDDLES;
      printf("L %ld\t",pDistance[LEFTS]);
      break;
    case MIDDLES:
      sendf("READ C=5,010000,000000 O=%x",SensorValue);
      pDistance[MIDDLES]=*(SensorValue+1);
      NextSensor = RIGHTS;
      printf("C %ld\t",pDistance[MIDDLES]);
      break;
    case RIGHTS:
      sendf("READ C=3,100000,000000 O=%x",SensorValue);
      pDistance[RIGHTS]=*SensorValue;
      NextSensor = LEFTS;
      printf("R %ld\t",pDistance[RIGHTS]);
      break;
  }
```

```
} /* ReadRight */

/*..........................................................
ReadLeft(pDistance) - reads alternately all three front sensor
values.
..........................................................*/
void ReadLeft(pDistance)
long int *pDistance;
{
static NextSensor = LEFTS;
long int SensorValue[20];

  switch(NextSensor){
    case LEFTS:
      sendf("READ C=5,100000,000000 O=%x",SensorValue);
      pDistance[LEFTS]=*SensorValue;
      NextSensor = MIDDLES;
      printf("L %ld\t",pDistance[LEFTS]);
      break;
    case MIDDLES:
      sendf("READ C=3,010000,000000 O=%x",SensorValue);
      pDistance[MIDDLES]=*(SensorValue+1);
      NextSensor = RIGHTS;
      printf("C %ld\t",pDistance[MIDDLES]);
      break;
    case RIGHTS:
      sendf("READ C=6,100000,000000 O=%x",SensorValue);
      pDistance[RIGHTS]=*SensorValue;
      NextSensor = LEFTS;
      printf("R %ld\t",pDistance[RIGHTS]);
      break;
  }

} /* ReadLeft */

/*..........................................................
InitMovements() - initialises and starts the motion process.
..........................................................*/
void InitMovements()
{
int i;
long int DevStat, time1;
char *TextLine;
extern short int DevNumb;
extern struct robotdata Robot;
extern Procs_To_Add tab1;
long int *get_dev_status(), timer();
long int send(), sendf();
long int init_procit(),add_procit();
void GetText();

  for (i=0;i<STATEDIM;i++) Offset[i]=0;

  DevStat=*get_dev_status(DevNumb);
  while(DevStat!=1){
    send("MOTN OFF\r");
    send("JSTK OFF\r");
    DevStat=*get_dev_status(DevNumb);
  }

  time1=timer();
  while(timer()<(time1+(long)100));         /* wait 1 second */

  open_port2();
  Report("\n\r");
  Report("\nCut the link and press <Enter> ...\n\r");
  GetText(TextLine);

  sendf("ERR A=%x",Error);
  send("MOTN ON\r");

  time1=timer();
  while(timer()<(time1+(long)100));         /* wait 1 second */
  send("MOTN FORWARD D=1 V=25\r"); /* get rid of MOTN ST bug */
  time1=timer();           /* after making a JSTK ON and OFF */
  while(timer()<(time1+(long)200));         /* wait 2 second */

  /* configure us-sensros */
  send("RNOD R=ON\r");

  Robot.Code = READY;
  Robot.Status = INIT;
  Robot.Direction = "STOPPED";
  Robot.Communic = INITIALISE;

  init_procit();
  while (add_procit(&tab1));

  while(Robot.Communic!=ACK);
  while(Robot.Code!=READY);
  Robot.Status = OK;

} /* InitMovements */

/*..............................................................
ReturnRobot() - return robot to start position.
..............................................................*/
void ReturnRobot()
{
long int time1;
char *TextLine;
extern struct robotdata Robot;
long int send();
long int timer();
void GetText();

  time1=timer();
  while(timer()<(time1+(long)100));         /* wait 1 second */

  send("MOTN OFF\r");

  time1=timer();
  while(timer()<(time1+(long)100));         /* wait 1 second */

  send("JSTK ON\r");

  Report("\nThe joystick is now switched on.\n\r");
  Report("\nPlease move the robot to the correct position\n");
  Report("\rand press <Enter> ...\n\r");
  GetText(TextLine);
  Report("\n\r");
  send("JSTK OFF\r");

  time1=timer();
  while(timer()<(time1+(long)100));         /* wait 1 second */

  send("MOTN ON\r");

  time1=timer();
  while(timer()<(time1+(long)100));         /* wait 1 second */
  send("MOTN FORWARD D=1 V=25\r"); /* get rid of MOTN ST bug */
  time1=timer();           /* after making a JSTK ON and OFF */
  while(timer()<(time1+(long)200));        /* wait 2 seconds */

} /* ReturnRobot */


/*...............................................................
TermMovements() - remove motion process.
...............................................................*/
void TermMovements()
{
long int time1;
char *TextLine;
extern struct robotdata Robot;
extern Procs_To_Remove tab2;
long int send();
long int rem_procit();
long int timer();
void GetText();

  while(Robot.Code!=READY);

  while(rem_procit(&tab2));

  time1=timer();
  while(timer()<(time1+(long)100));         /* wait 1 second */

  send("RNOD R=OF\r");
  send("SERV OFF\r");
  send("MOTN OFF\r");
  send("ERR\r");

  time1=timer();
  while(timer()<(time1+(long)100));        /* wait 1 secondl */

  send("JSTK ON\r");

  Report("\nThe experience has been completed.\n\r");
  Report("\nThe joystick is now switched on.\n\r");
  Report("\nPlease move the robot to the initial position.\n");
  Report("\r\nTo see the results, restore the link\n\r");
  Report("and press <Enter> ...\n\r");
  GetText(TextLine);

  send("JSTK OFF\r");

} /* TermMovements */

/*...............................................................
GenerateSonarAlarm(State,Action)- generates a sonar alarm for
simulation.
```

```
...........................................................*/
void GenerateSonarAlarm(State,Action)
pstate State;
action Action;
{
int i,x,y;
extern boolean SonarAlarm;

  /* left border */
  /*for(i=1;i<(YMAX-1);i++)
    if(State[X]==1 && State[Y]==i && Action==LEFT)
      SonarAlarm=TRUE;*/

  /* right border */
  for(i=1;i<(YMAX-1);i++)
    if(State[X]==(XMAX-2) && State[Y]==i && Action==RIGHT)
      SonarAlarm=TRUE;

  /* upper border */
  /*for(i=1;i<(XMAX-1);i++)
    if(State[X]==i && State[Y]==1 && Action==UP)
      SonarAlarm=TRUE;*/

  /* lower border */
  for(i=1;i<(XMAX-1);i++)
    if(State[X]==i && State[Y]==(YMAX-2) && Action==DOWN)
      SonarAlarm=TRUE;

  /* obstacle at */ x = 2; y = 1;
  /*
  if(State[X]==(x-1) && State[Y]==y && Action==RIGHT)
    SonarAlarm=TRUE;
  if(State[X]==x && State[Y]==(y-1) && Action==DOWN)
    SonarAlarm=TRUE;
  if(State[X]==x && State[Y]==(y+1) && Action==UP)
    SonarAlarm=TRUE;
  if(State[X]==(x+1) && State[Y]==y && Action==LEFT)
    SonarAlarm=TRUE;
  */

} /* GenerateSonarAlarm */

/*...........................................................
Print Status(Text)- this routine prints the actual value of the
variables:
Robot.Communic, Robot.Direction,  Robot.Code and Robot.Status.
To use for debugging.
...........................................................*/
void PrintStatus(Text)
char Text[];
{
char *mstr1 = "";
extern struct robotdata Robot;
void Report();

/*  printf("\nComm\t Motion\t Code\t Stat\t %s\n",Text);*/

  switch(Robot.Communic){
    case 0: Report("CLEAR\t"); break;
    case 1: Report("NEW_COM\t"); break;
    case 2: Report("ACK\t"); break;
    case 3: Report("EM_STO\t"); break;
    case 4: Report("INIT\t"); break;
  }
  sprintf(mstr1,"%c%c%c%c",Robot.Direction[0],
    Robot.Direction[1], Robot.Direction[2],Robot.Direction[3]);
  Report(mstr1);
  Report("\t");
  switch(Robot.Code){
    case 0: Report("EXECUT\t"); break;
    case 1: Report("READY\t"); break;
    case 2: Report("ABORT\t"); break;
  }
  switch(Robot.Status){
    case 0: Report("INIT"); break;
    case 1: Report("OK"); break;
    case 2: Report("OBSTAC"); break;
    case 3: Report("TROUBL"); break;
  }
  Report("\n\r");

} /* PrintStatus */

/*...............................................................
Report(TextLine)- writes the sring on port 2.
...............................................................*/
void Report(TextLine)
char *TextLine;
{
struct serial2_block_IO CtrlBlock;
struct serial2_block_test TestBlock;

  while(test_output_serial2(&TestBlock)==status2_BUSY);
  write_serial2(TextLine,&CtrlBlock);

} /* Report */

/*...............................................................
GetText(TextLine)- reads the sring from port 2.
...............................................................*/
void GetText(TextLine)
char *TextLine;
{
struct serial2_block_IO CtrlBlock;
struct serial2_block_test TestBlock;
short int size;

  /*test_input_serial2(&TestBlock);
  while(TestBlock.length==0)
    test_input_serial2(&TestBlock);
  while(!isspace(TextLine[0]))
  while(test_output_serial2(&TestBlock)==status2_BUSY);*/

  /*do{
    test_input_serial2(&TestBlock);
    if(TestBlock.length>4){
      read_serial2(TextLine,&CtrlBlock,4);
      printf("\n%s\n",TextLine);
      if(strcmp(TextLine[0],"\r")) break;
    }
  }while(TRUE);*/
  while(test_output_serial2(&TestBlock)==status2_PERFOUT);

  while(test_input_serial2(&TestBlock)!=status2_ENDDELIM);

  size = TestBlock.length;
  read_serial2(TextLine,&CtrlBlock,size);
  printf("\n%s\n",TextLine);

} /* GetTextLine */
```

C.7 File README (Operating Instructions)

```
DESCRIPTION AND OPERATING INSTRUCTIONS FOR motion.tests

This directory contains the files needed to test the functions
and develop the module Movements.c. Here you can test the
interfaces used between DynAlg and Movements to make the robot
execute simple translations and rotations or evasive actions,
make sonar readings, etc.

The files are:
- DynAlg.h
- passear.c
- Movements.h
- Movements.c
- Makefile


You first have to complete the main routine in passear.c.
See the examples in the comments of the main routine.
You can make rotations with angles between -360 and +360,
translation, scannings to generate an sonar alarm or test the
communication via modem, or test new created functions.

You compile, download and start the program in the same way
as explained in 'experiments'.

>kim{weiseral}5: make load
>cc -I/home/local/asdp/include -target sun3 -c -m68020 -f68881
>-O passear.c
>/usr/arch/sun4/commonR4.0/crossbin/target sun3 ld -S -u start
>-e start -N -T 20000 -o passear \
>/home/local/asdp/lib/sunfp/start.o \
>passear.o \
>/home/local/asdp/lib/sunfp/libcalbh.a  /home/local/asdp/lib/su
>nfp/libcalbl.a  /home/local/asdp/lib/sunfp/libcpro.a  /home/lo
>cal/asdp/lib/sunfp/libcstd.a  /home/local/asdp/lib/sunfp/libio
>.a  -lc \
>     /home/local/xcomp/sun4R4.0/sun3R4.0/usr/lib/f68881/libm.a
>  /home/local/xcomp/sun4R4.0/sun3R4.0/usr/lib/libc.a
>/home/local/asdp/bin/sformat -o passear.mx passear
>/home/local/asdp/bin/sload -d /dev/ttya passear.mx
>kim{weiseral}6:
>kim{weiseral}6: tip -9600 /dev/ttya
>connected
>
>Robuter 5.2>
>Robuter 5.2>g 20000
```

C.8 File passear.c

```
/*..............................................................
passear.c - routine to make the first experiments of robot
control before running DynAlg.
...............................................................
Unit:
File:  passear.c
Date:  19/04/95
Author:  Alex Weiser
Version: 1.0
Revisions:
Comments:  This unit contains all the control of the movements
           necessary for the DynAlg. The next step is to use
           this unit together with DynAlg simulation to run the
           algorithm on the real robot.

Functions:  The function that are provided for the user are:
            InitMovements()
            Rotate(Angle)
            Translate()
            TermMovements()
..............................................................*/

#include "libcalbh.h"
#include "libcalbl.h"
#include "libcpro.h"
#include "libcstd.h"
#include "libio.h"
#include "DynAlg.h"
#include "Movements.h"
#include "Movements.c"

Procs_To_Add tab1 = {1,{MotionProcess,10,1,0}};
Procs_To_Remove tab2 = {1,MotionProcess};
main()
{
int i=0;
char c;
boolean Translate(),Rotate();
void InitMovements(),TermMovemets(),Report();
long int time1, timer();

  /*Initialise*/

  printf("\n");
  printf("You will never get it man!! ");

  scanf("%c ",&c);
  printf("\n");



   /*
    for(i=0;i<6;i++)           how to make sonar readings
      while(!Scan(LEFTSD,MIN_DIST1));*/


  InitMovements(); /* necessary to make a translation or
                      rotation                            */
  do{              /* example of motions */
    Rotate(0);
    Translate();
  }

  while(i++<10)
    while(Translate());

  Rotate(-90);
  Translate();
  while(Translate());

  while(Rotate(90)
      Translate();

    /*Report("Hello");    modem communication               */

  do{       /*   example how to make a loop using the timer */
    time1=timer();
    while(timer()<(time1+(long)500));
    printf("Again?? (1/0) ");
    scanf("%d ",&i);
    printf("\n");

  }while(i);


  TermMovements(); /* necessary to terminate the motion
                                                process */
}
```

Appendix D

Results Using the Robot

D.1 Maps

D.1.1 Experiment with $\alpha = 1000$, Seed=1 and 50 Hypothetical Steps

```
REAL EXPERIENCE 1
     #           #
 #  22   #   1   5   #
 #  20   #   #   4   3
 #  21  12  15   7   8   #
     #   #   #   #   #
 118 steps

REAL EXPERIENCE 2
     #           #
 #   5   #   1   1   #
 #   1   #   #   1
 #   3   2   1   1       #
     #   #   #   #   #
 16 steps

REAL EXPERIENCE 3
     #           #
 #   1   #   1   1   #
 #   1   #   #   1
 #   1   1   1   1       #
     #   #   #   #   #
 9 steps

REAL EXPERIENCE 4
     #           #
 #       #   1   1   #
 #   1   #   #   1
 #   1   1   1   1       #
     #   #   #   #   #
 8 steps

REAL EXPERIENCE 5
     #           #
 #       #   1   1   #
 #   1   #   #   1
 #   1   1   1   1       #
     #   #   #   #   #
 8 steps
```

D.1.2 Experiment with $\alpha = 1000$, Seed=8 and 50 Hypothetical Steps

```
Optimal path found in 3 trials.
Need 8 steps to reach goal.

REAL EXPERIENCE 1
     #
 #   3       1   1
 #  11   #   #   1
 #   5   2   2   2
                 #
 28 steps 21 calls 16 motions

REAL EXPERIENCE 2
     #
 #   8   #   1   1
 #   4   #   #   1
 #   1   1   1   1
                 #
 19 steps 11 calls 12 motions

REAL EXPERIENCE 3
     #
 #       #   1   1
 #   1   #   #   1
 #   1   1   1   1
                 #
 8 steps 8 calls 8 motions

REAL EXPERIENCE 4
     #
 #       #   1   1
 #   1   #   #   1
 #   1   1   1   1
                 #
 8 steps 8 calls 8 motions
```

D.2 Policy and Evaluation Tables

D.2.1 Experiment with $\alpha = 1000$, Seed=1 and 50 Hypothetical Steps

```
Evaluation Table

  0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000

  0.0000   0.4780   0.0000   0.0000   1.0000   0.0000   0.0000

  0.0000   0.5314   0.0000   0.0000   0.9000   0.8100   0.0000

  0.0000   0.5905   0.6561   0.7290   0.8100   0.7290   0.0000

  0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000
```

```
Policy Table beginning by x=0

+ -    - + -    - + -    - + -    - + -    - + -    - + -    - +
|  25.  |  25.  |  25.  |  25.  |  25.  |  25.  |  25.  |
  25. 25.  25. 25.  25. 25.  25. 25.  25. 25.  25. 25.  25. 25.
|  25.  |  25.  |  25.  |  25.  |  25.  |  25.  |  25.  |
+ -    - + -    - + -    - + -    - + -    - + -    - + -    - +
|  25.  |   1.  |  25.  |  25.  |   0.  |  25.  |  25.  |
  25. 25.   1.  1.  25. 25.  25. 25. 100.  0.  25. 25.  25. 25.
|  25.  |  98.  |  25.  |  25.  |   0.  |  25.  |  25.  |
+ -    - + -    - + -    - + -    - + -    - + -    - + -    - +
|  25.  |   0.  |  25.  |  25.  | 100.  |   0.  |  25.  |
  25. 25.   0.  0.  25. 25.  25. 25.   0.  0. 100.  0.  25. 25.
|  25.  |  99.  |  25.  |  25.  |   0.  |   0.  |  25.  |
+ -    - + -    - + -    - + -    - + -    - + -    - + -    - +
|  25.  |   0.  |   0.  |   0.  | 100.  |  91.  |  25.  |
  25. 25.   0. 99.   0.100.   0.100.   0.  0.   8.  1.  25. 25.
|  25.  |   0.  |   0.  |   0.  |   0.  |   1.  |  25.  |
+ -    - + -    - + -    - + -    - + -    - + -    - + -    - +
|  25.  |  25.  |  25.  |  25.  |  25.  |  25.  |  25.  |
  25. 25.  25. 25.  25. 25.  25. 25.  25. 25.  25. 25.  25. 25.
|  25.  |  25.  |  25.  |  25.  |  25.  |  25.  |  25.  |
+ -    - + -    - + -    - + -    - + -    - + -    - + -    - +
```

D.2.2 Experiment with $\alpha = 1000$, Seed=8 and 50 Hypothetical Steps

```
Optimal path found in 3 trials.
Need 8 steps to reach goal.

Evaluation Table

 0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000

 0.0000  0.4777  0.0000  0.0000  1.0000  0.0000  0.0000  0.0000

 0.0000  0.5311  0.0000  0.0000  0.9000  0.0000  0.0000  0.0000

 0.0000  0.5902  0.6559  0.7290  0.8100  0.0000  0.0000  0.0000

 0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000

Policy Table
+ -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - +
|  25.  |  25.  |  25.  |  25.  |  25.  |  25.  |  25.  |  25.  |
 25. 25. 25. 25. 25. 25. 25. 25. 25. 25. 25. 25. 25. 25. 25. 25.
|  25.  |  25.  |  25.  |  25.  |  25.  |  25.  |  25.  |  25.  |
+ -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - +
|  25.  |   1.  |  25.  |  25.  |   0.  |  25.  |  25.  |  25.  |
 25. 25.  1.  1. 25. 25. 25. 25. 100. 0. 25. 25. 25. 25. 25. 25.
|  25.  |  98.  |  25.  |  25.  |   0.  |  25.  |  25.  |  25.  |
+ -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - +
|  25.  |   0.  |  25.  |  25.  | 100.  |  25.  |  25.  |  25.  |
 25. 25.  0.  0. 25. 25. 25. 25.  0.  0. 25. 25. 25. 25. 25. 25.
|  25.  |  99.  |  25.  |  25.  |   0.  |  25.  |  25.  |  25.  |
+ -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - +
|  25.  |   0.  |   0.  |   0.  |  100. |  25.  |  25.  |  25.  |
 25. 25.  0.100.  0.100.  0.100.  0.  0. 25. 25. 25. 25. 25. 25.
|  25.  |   0.  |   0.  |   0.  |   0.  |  25.  |  25.  |  25.  |
+ -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - +
|  25.  |  25.  |  25.  |  25.  |  25.  |  25.  |  25.  |  25.  |
 25. 25. 25. 25. 25. 25. 25. 25. 25. 25. 25. 25. 25. 25. 25. 25.
|  25.  |  25.  |  25.  |  25.  |  25.  |  25.  |  25.  |  25.  |
+ -   - + -   - + -   - + -   - + -   - + -   - + -   - + -   - +
```