Αγγελόπουλος Χρήστος

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Υλοποίηση πρακτόρων σε πολυπρακτορικό σύστημα

Επιβλέπουσα: Δασκαλοπούλου Ασπασία
Συνεπιβλέπων: Ακρίτας Αλκιβιάδης

Τ. Μ. Η. Υ. Τ. Δ ΟΚΤΩΒΡΙΟΣ 2010

ΠΕΡΙΕΧΟΜΕΝΑ **ΣΕΛΙΔΑ**

Περίληψη

Σκοπός της παρούσας πτυχιακής εργασίας είναι η προγραμματιστική προσομοίωση ενός πολυπρακτορικού συστήματος επίλυσης του προβλήματος Tileworld σε γλώσσα C. Επιπλέον, η συλλογή στατιστικών στοιχείων προς εξέταση της συμπεριφοράς των πρακτόρων μέσω αλλαγής συγκεκριμένων μεταβλητών.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: *Tileworld, πράκτορας, C, πολυπρακτορικό σύστημα*

Περιγραφή προβλήματος

Έστω ένας πεπερασμένος διακριτός χώρος. Κάθε μέρος του χώρου αυτού μπορεί να είναι κενό (ή διάδρομος), τούβλο, τρύπα ή εμπόδιο. Στον κόσμο αυτό υπάρχουν πράκτορες, ενεργούσες οντότητες δηλαδή, οι οποίες αλληλεπιδρούν με τον κόσμο. Με δυνατότητα προοπτικής τεσσάρων κατευθύνσεων (πάνω, κάτω, αριστερά, δεξιά) εντοπίζουν τα τούβλα, τα παίρνουν και προσπαθούν αφού βρουν κάποια τρύπα να την γεμίσουν με ένα απ' αυτά, αποφεύγοντας τα εμπόδια στη διαδρομή τους.

Προοπτικές:

- Το περιβάλλον του Tileworld είναι δυναμικό: ξεκινά σε μια τυχαία γεννημένη κατάσταση και μεταβάλλεται με την πάροδο του χρόνου σε διακριτά βήματα, καθώς εμφανίζονται και εξαφανίζονται τυχαία κενά. Ο προγραμματιστής του περιβάλλοντος μπορεί να θέσει διάφορες παραμέτρους (τη συχνότητα εμφάνισης/εξαφάνισης των κενών, τούβλων και εμποδίων).

- Η απόδοση ενός πράκτορα μετριέται τρέχοντας το περιβάλλον για ένα προκαθορισμένο αριθμό βημάτων και μετρώντας τον αριθμό των κενών που καταφέρνει να γεμίσει ο πράκτορας στο διάστημα αυτό. Η απόδοση του πράκτορα στη διάρκεια μιας εκτέλεσης r ορίζεται ως:

$$u(r) = \frac{\#\, holes\ \_\ filled\ \_\ in\ \_\ r}{\#\, holes\ \_\ appeared\ \_\ in\ \_\ r}$$

- Το αποτέλεσμα είναι μια απόδοση στο διάστημα [0,1] όπου 0 = ο πράκτορας δεν κατάφερε να γεμίσει καμία τρύπα και 1= ο πράκτορας γέμισε κάθε κενό που εμφανίστηκε.
- Οι πράκτορες μπορεί να επικοινωνούν μεταξύ τους ή όχι.

- Η ευαισθησία των πρακτόρων στις αλλαγές του κόσμου είναι μια ενδιαφέρουσα προοπτική ως προς το πόσο ευφυείς είναι. Για παράδειγμα, αν θεωρείται σημαντικό από τον προγραμματιστή προτίμηση στα εγγύτερα τούβλα και ο πράκτορας επιλέξει κάποιο νεοεμφανισθέν κοντινότερο τούβλο απ' αυτό που αρχικά νομίζει ότι τον καλύπτει, θεωρείται ευφυής ενέργεια.

Ζητήματα υλοποίησης

Περιληπτικά

- Μοντελοποίηση και διακριτοποίηση του κόσμου
- Κριτήρια εύρεσης αντικειμένων (*εγγύτητα, προσβασιμότητα κλπ*)
- Επικοινωνία πρακτόρων
- Ρύθμιση ευαισθησίας σε αλλαγές
- Συνθήκες τερματισμού
- Εύρεση μονοπατιού προς τα αντικείμενα
- Εξέταση «κακών» υποπεριπτώσεων (*π. χ λιγότερα τούβλα από τρύπες*)
- Συγχρονισμός – ανοχή προγράμματος για ταυτόχρονο τρέξιμο πρακτόρων και πρακτόρων – κόσμου

Υλοποίηση αναλυτικά

Ο κόσμος αποτελείται από έναν πίνακα ακεραίων διάστασης M x N, οι οποίες καθορίζονται από μακροεντολές πριν τη μεταγλώττιση, όπου χρησιμοποιούμε:

- 0 για το διάδρομο
- 1 για τα τούβλα
- 2 για τα εμπόδια
- 3 για τις τρύπες
- 4 για τις γεμισμένες τρύπες
- 8 για τη θέση του πράκτορα

Αρχικοποιείται κατά προαίρεση του χρήστη στατικά, με ανάλογη συνάρτηση όπου φαίνονται με μη-επαναληπτικό τρόπο οι συντεταγμένες για όλα τα στοιχεία πλην της θέσης του πράκτορα που τίθεται σε άλλο τμήμα του προγράμματος και του διαδρόμου (εξ ορισμού αρχικά όλα τα στοιχεία θεωρούνται διάδρομος).

Για εύρεση επόμενης τρύπας ή τούβλου χρησιμοποιείται συνάρτηση διαπέρασης του κόσμου που επιστρέφει το πρώτο διαθέσιμο στοιχείο.

Ο κόσμος αλλάζει ως εξής. Καλείται μια συνάρτηση, που τρέχει παράλληλα με τους πράκτορες για συγκεκριμένο αριθμό φορών που προκαθορίζονται από το χρήστη, αποφασίζει τυχαία μία από πεπερασμένο πλήθος ενεργειών. Συγκεκριμένα εμφάνιση/εξαφάνιση τούβλου, τρύπας και εμποδίου και καμία ενέργεια.

Στη συνέχεια βρίσκεται το πρώτο διαθέσιμο στοιχείο που μπορεί να την υποστεί (τούβλο αν η ενέργεια είναι εξαφάνιση τούβλου) και η ενέργεια εκτελείται αν είναι δυνατό. Το τελευταίο εξαρτάται από το αν έχει προλάβει ο/οι πράκτορας/πράκτορες να αλλοιώσει πριν τον κόσμο, δηλαδή θέμα συγχρονισμού μεταξύ των δύο οντοτήτων.

Ο πράκτορας, διαθέτει ένα αναγνωριστικό για να γίνεται φανερό ποιος τρέχει κάθε φορά και τη βαθμολογία του. Υπάρχουν δυο μονοδιάστατοι πίνακες ακεραίων για το καθένα μήκους ίσου με τον αριθμό των πρακτόρων.

Κάθε φορά τρέχει μόνο ένας πράκτορας στο σύστημα ο οποίος κλειδώνει τους υπόλοιπους σε έναν δυαδικό σηματοφόρο, για να εξαλειφθούν ανεπιθύμητες συνθήκες ανταγωνισμού. Ο πράκτορας ψάχνει για το πρώτο διαθέσιμο τούβλο στον κόσμο, κινείται προς αυτό και το παίρνει από την θέση του. Ακολούθως,

κινείται προς την πρώτη ευρεθείσα τρύπα και τη γεμίζει. Αν δε βρει κάποια τρύπα, το τούβλο καταστρέφεται. Εκτελείται όσο υπάρχουν διαθέσιμα τούβλα ή τρύπες στο σύστημα.

Η κίνηση του πράκτορα αποτελεί από μόνη της ένα πρόβλημα (βλ. επίλυση λαβυρίνθου ή maze solving). Είθισται να χρησιμοποιούνται συγκεκριμένοι αλγόριθμοι για αυτή μεταξύ των οποίων οι Wall follower, Pledge, Trımaux, Dead-end filling και Shortest path. Οι τελευταίοι εμπεριέχουν τεχνικές τυφλής αναζήτησης (DFS, BFS, A* κλπ). Το συγκεκριμένο πρόγραμμα έγινε με τον BFS, ο οποίος κάνει τα εξής βήματα.

- Βάζει τον κάθε κόμβο προς εξέταση σε μια στοίβα (ή οποιαδήποτε LIFO δομή, εν προκειμένω στην κεφαλή μιας ουράς). Άδεια ουρά ισοδυναμεί με αδυναμία εύρεσης λύσης.

- Τον βγάζει από την ουρά για εξέταση. Αν είναι λύση τον κρατάει, αν όχι βάζει τα παιδιά του στην ουρά *(αναζήτηση κατά πλάτος)*. Εν προκειμένω, αν (x, y) είναι το τετράγωνο που βρίσκεται ο πράκτορας τοποθετούνται τα (x-1,y) , (x,y-1), (x+1,y), (x+1, y+1) εφόσον είναι έγκυρα τετράγωνα του πίνακα-κόσμου.

- Κάθε κόμβος που εξετάζεται τοποθετείται στο σύνολο closure ώστε πρώτα γίνεται έλεγχος εαν υπάρχει εκεί. Αν όχι, ο αλγόριθμος διατρέχεται απ την αρχή.

Ο συντονισμός των παραπάνω ενεργειών, γίνεται ως εξής. Αρχικοποιείται ο κόσμος, τοποθετούνται οι πράκτορες και ύστερα τρέχουν ως νήματα μαζί με τη συνάρτηση τυχαιοποίησης του κόσμου. Αυτή συνήθως εκτελεί πρώτα ορισμένες (συνηθέστερα μία ή δυο) ενέργειες πριν οι πράκτορες αρχίσουν. Ο κόσμος τερματίζει την ζωή του ύστερα από προκαθορισμένο αριθμό ενεργειών και ο πράκτορας όταν εξαφανιστούν τα τούβλα ή οι τρύπες.

Παραδοχές

- Ο πράκτορας μπορεί να περπατάει και πάνω από τούβλα χωρίς να τα μετακινεί
- Ο πράκτορας δεν επικοινωνεί καθόλου με τους υπόλοιπους και όχι άμεσα τουλάχιστον με το περιβάλλον και τον κόσμο. Βασίζεται σε λύσεις τύπου brute-force αφού επιλέγει πάντα την πρώτη διαθέσιμη ενέργεια.
- Οι πράκτορες τοποθετούνται με συγκεκριμένο τρόπο στον κόσμο, χάριν απλούστευσης (για τοποθέτηση πολλών με βρόχο επανάληψης). Ο πρώτος στη θέση (2,1) και οι επόμενοι ένα τετράγωνο διαγωνίως ((3,2) , (4,3), …). Πρέπει να λαμβάνεται μέριμνα ώστε ο τελευταίος πράκτορας να μην ξεπερνά τα όρια του πίνακα – κόσμου.
- Οποιαδήποτε αλλαγή του κόσμου επηρεάζει τις ενέργειες του πράκτορα όσο αυτός τρέχει. Οι μετέπειτα αλλαγές δεν έχουν νόημα.
- Ως φόρτο ορίζεται το ελάχιστο των τούβλων ή των τρυπών δια τον αρ. των πρακτόρων.
- Η αλλαγή αριθμού τρυπών στο σύστημα αλλάζει τον παρονομαστή στη συνάρτηση βαθμολογίας του πράκτορα, εφόσον υπάρχουν ακόμα τούβλα (σαν αύξηση των εναλλακτικών ή εν μέρει του φόρτου του πράκτορα).
- Το άθροισμα βαθμολογιών των πρακτόρων δεν πρέπει είναι απαραιτήτως 1. Για παράδειγμα αν 3 πράκτορες τελείωσαν τη ζωή τους γεμίζοντας ισάριθμες τρύπες (έστω 2 ο καθένας από 8 συνολικά) θα έχουν άθροισμα 2/8+2/8+2/8 = 0,75. Αν ένας τέταρτος δεν προλάβει να γεμίσει τη μία από τις 2 εναπομείνασες τρύπες ή εμφανιστεί τρίτη τρύπα στο σύστημα ενώ υπάρχουν τούβλα, η βαθμολογία του θα είναι 1/7=0,141. Άρα το συνολικό άθροισμα θα είναι 0,891 < 1.
- Στη συνάρτηση κάθε πράκτορα εισήχθη μικρή καθυστέρηση (διπλός βρόχος) για να είναι μετρήσιμοι οι υπό κανονικές συνθήκες μικροί χρόνοι που απαιτούνται για διεκπεραίωση της εργασίας κάθε πράκτορα.
- Παρότι χρησιμοποιούνται αρκετά γεννήτριες τυχαίων αριθμών, η έκβαση του προγράμματος είναι σχεδόν η ίδια, λόγω της ψευδοτυχαιότητας.
- Η πεπερασμένη χρονική διάρκεια του προγράμματος διασφαλίζεται με μη-επανατοποθετήσιμα τούβλα, πεπερασμένο αριθμό ενεργειών κόσμου και ύπαρξη τερματικής συνθήκης στον πράκτορα.
- Η γεμισμένη τρύπα λειτουργεί προσωρινά και ως θέση πράκτορα, γιατί η αναζήτηση για την επόμενη ενέργεια (ή η αρχική θέση της νέας διαδρομής από τον BFS) έχει αυτή ως αρχή.
- Χάριν απλούστευσης στους χρόνους διεκπεραίωσης, θεωρούμε ότι όλοι οι πράκτορες μπαίνουν στο σύστημα την στιγμή 0.
- Οι εκτελέσεις αριθμούνται κατά αύξουσα σειρά συναρτήσει του αριθμού πρακτόρων που συμμετέχουν σ αυτές.
- Υπάρχει ανταγωνισμός μεταξύ των πρακτόρων με κάποιους εξ αυτών να μένουν ανενεργοί σε κάποιες εκτελέσεις, καθώς δεν θα προλάβουν (λόγω του κλειδώματος και του ταυτόχρονου τρεξίματος του κόσμου) να διεκπεραιώσουν κάποιες ενέργειες. Αναμένεται το ποσοστό τους να αυξάνει ανάλογα με το συνολικό αριθμό πρακτόρων. Εν μέρει δείκτη ανενεργών (ή κατ’ ακρίβειαν ισομερισμού του φόρτου) πρακτόρων αποτελούν η διασπορά και η τυπική απόκλιση της βαθμολογίας.

- Οι μέσοι όροι και τα άλλα στατιστικά προκύπτουν από το σύνολο των εκτελέσεων ανά εκδοχή.

Αναλυτικά σχολιασμένος κώδικας

agent.h

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include "tileworld.h"

void ag_init();

int is_filled(int i,int j);

void *agent(struct coord *dimz)
```

agent.c

```
/*Arxeio synarthsewn praktora*/



#include <pthread.h>    /* POSIX Threads */
#include <stdio.h>
#include <stdlib.h>
#include "tileworld.h"
#include "mybfs.h"
#include "queue.h"
#include <time.h>
```

```
struct coord dims;


int has_brick; // 0 : praktoras den exei touvlo, 1: exei


int holess[AGENTS]; // Ari8mos trypwn pou gemise o praktoras (gia to score)

float ag_score[AGENTS];


clock_t start,end;

double total;     //Gia ektypwsh score tou praktora


// Shmatoforos pou prostateuei tis energeies tou praktora

pthread_mutex_t mutex=PTHREAD_MUTEX_INITIALIZER;


int j;

void ag_init()

{

/*O praktoras arxika den exei touvlo, to score tou einai mhden kai oi gemates
trypes mhden*/

      has_brick = 0;

      ag_score[ag_id]=(float)0;

      holess[ag_id]=0;

}




void *agent(struct id_coord *dimz)

{

      /*syntetagmenes arxikhs & telikhs 8eshs :

        a)praktora        b)touvlou  c)trypas opou 8a mpei*/

      int x_init,y_init, mid_x,mid_y, final_x,final_y;

      int i,ag_id;


/*mesh diadromhs(mid)h euresh touvlou, telos(final) h trypa opws epistrefontai apo
  BFS*/
```

```
struct coord mid,final;

x_init=dimz->x;y_init=dimz->y;ag_id=dimz->id;

ag_init();
start = clock();



//Oso yparxoun trypes ‘
while(count_data(HOLE) && count_data(BRICK))
{
      sleep(random()%AGENTS);

      pthread_mutex_lock(&mutex);
      /*Metavash se touvlo - teleutaio orisma 0 giati den psaxnoume gia
      trypes*/
      mid=return_data(BRICK);
      my_bfs(ag_id,x_init,y_init,mid.x,mid.y,0);

      //Se periptwsh kakou sygxronismou psakse gia neo touvlo
      if(board[mid.x][mid.y]!=BRICK)
      {pthread_mutex_unlock(&mutex);continue;}
      else
      {
            //O praktoras "shkwnei" to touvlo...
            has_brick=1;
            //...kai to kommati diadromou tou touvlou periexei auton...
            board[mid.x][mid.y]=AGNT_POS;
            printf("(AG-%d) : Brick (%d,%d)",ag_id,mid.x,mid.y);
            //...enw to prohgoumeno ginetai diadromos
            board[x_init][y_init]=BLOCK;
            final= return_data(HOLE);
```

```
/*An den yparxei dia8esimh trypa katastrefetai to touvlo (gia th syn8hkh
termatismou)*/
                  if(final.x==-5)
                  {
                        printf(" destroyed\n");
                        //kai ase tous allous praktores na perasoun an yparxoun
                        pthread_mutex_unlock(&mutex);continue;
                  }
                  else
                  {
                        //Metavash se trypa
                        my_bfs(ag_id,mid.x,mid.y,final.x,final.y,1);
                        //O praktoras efyge apo kei pou htan
                        board[mid.x][mid.y]=BLOCK;
                        printf(" at (%d,%d)",final.x,final.y);
                        //kai phge sth gemath trypa
                        board[final.x][final.y]=FILLED;
                        ag_score[ag_id]++;holess[ag_id]++;
                        has_brick=0;
                        printf(" ,%d/%d totally\n\n",holess[ag_id],init_holes);

                        //H arxikh 8esh tou praktora gia thn epomenh anazhthsh
                        x_init=final.x;y_init=final.y;
                        pthread_mutex_unlock(&mutex);
                  }
            }
      }

      //Eisagwgh ka8ysterhshs gia na einai metrhsimoi oi xronoi twn praktorwn
      for(i=0;i<4000;i++){for(j=0;j<3000;j++){}}
      end = clock();
```

```
            /*Xronos se xtypous rologiou, diairesh gia stroggylopoihsh se logika
            megalous ari8mous (taksh mege8ous monadas)*/

            total = ((double)( end - start ))/10000;

printf("\n\t---(AG-%d) time
:%.2f,\tScore:%.3f\t\n\n",ag_id,total,ag_score[ag_id]/init_holes);

      }
```

mybfs.h

```
            #include <stdio.h>

            #include <stdlib.h>

            #include <math.h>

            #include "queue.h"



      struct QueueValue {

      int x;

      int y;

      }*dims1;


      Queue *front; /*to metwpo anazhthshs*/


      struct QueueValue closure[50];


      int find_holes;


      void init_closure(); /*arxikopoihsh tou synolou closure*/


      int is_on_closure(int x1,int y1); /*anazhthsh an to stoixeio me syntetagmenes
      x1,y1 anhkei ston pinaka*/
```

```
void put_chld(int x_chld,int y_chld);     /*perasma do8entos komvou sto metwpo*/


int check_children(int x,int y); /*elegxos gia yparkto index*/


struct QueueValue put_children(int x,int y,int search_holes); /*eisagwgh twn
paidiwn sto front*/


void my_bfs(int iid,int init_x,int init_y,int final_x,int final_y,int data);
/*ylopoihsh tou bfs me perasma arxikhs kai telikhs 8eshs*/
```

mybfs.c

```
  /*Ylopoihsh tou algori8mou BFS */

      #include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include "queue.h"

#include "mybfs.h"

#include "tileworld.h"


int clos_ind;     /*deikths gia ton pinaka closure*/


int prev_x,prev_y;      /*voh8htikes gia bfs*/


void init_closure()

{

      clos_ind=0;

      int i;

      for(i=0;i<50;i++)

      {closure[i].x=-20;closure[i].y=-20;}

}
```

```
int is_on_closure(int x1,int y1)
{
	int i;
	for(i=0;i<50;i++)
	{
		if((closure[i].x==x1)&&(closure[i].y==y1))
		{return 1;}
	}
	return -1;
}



int check_children(int x,int y)
{
	return((x>=0)&&(x<M)&&(y>=0)&&(y<N));
}

void put_chld(int x_chld,int y_chld)
{
		dims1 = (struct QueueValue *)malloc(sizeof(struct QueueValue));
		dims1->x=x_chld;dims1->y=y_chld;


		queue_push_head(front,dims1);
}

struct QueueValue put_children(int x,int y,int search_holes)
{
	struct QueueValue ret_val,*tmp;

	//yparkto index && oxi empodio
	if(check_children(x-1,y))
```

```
{     //perpathma panw apo diadromo kai touvla
      if((board[x-1][y]==BLOCK)||(board[x-1][y]==BRICK))
      {put_chld(x-1,y);}

      if((search_holes ==BRICK)&&(board[x-1][y]==HOLE))
      {put_chld(x-1,y);}
}



if(check_children(x+1,y))
{
      if((board[x+1][y]==BLOCK)||(board[x+1][y]==BRICK))
      {put_chld(x+1,y);}

      if((search_holes ==BRICK)&&(board[x+1][y]==HOLE))
      {put_chld(x+1,y);}
}

if(check_children(x,y-1))
{
      if((board[x][y-1]==BLOCK)||(board[x][y-1]==BRICK))
      {put_chld(x,y-1);}

      if((search_holes ==BRICK)&&(board[x][y-1]==HOLE))
      {put_chld(x,y-1);}
}

if(check_children(x,y+1))
{
      if((board[x][y+1]==BLOCK)||(board[x][y+1]==BRICK))
      {put_chld(x,y+1);}

      if((search_holes ==BRICK)&&(board[x][y+1]==HOLE))
```

```
            {put_chld(x,y+1);}
      }
}


void my_bfs(int iid,int init_x,int init_y,int final_x,int final_y,int
search_holes)
{
      struct QueueValue *dims1_ret_val;
      struct QueueValue *tmp;

      int ret_x,ret_y;
      int i;

      init_closure();
      front=queue_new();

      //1. Eisagwgh sthn oura
      dims1 = (struct QueueValue *)malloc(sizeof(struct QueueValue));
      dims1->x=init_x;dims1->y=init_y;
      queue_push_head(front,dims1);
      prev_x = init_x;prev_y=init_y;

      while(1)
      {
            if(queue_is_empty(front))
            {return;}
            //2. Eksagwgh tou prwtou komvou
            dims1_ret_val=queue_pop_head(front);

            ret_x=dims1_ret_val->x;
            ret_y=dims1_ret_val->y;

            //3.An exei eksetastei hdh des to epomeno
```

```
                if(is_on_closure(ret_x,ret_y)==1)
                {continue;}
                else
                {
                      //4a.An vre8ei diadromos o praktoras kineitai
                      if(board[ret_x][ret_y]==BLOCK)
                      {
                            //kai an hr8e apo kapou, sthn arxikh 8esh tou ginetai
    diadromos
                            if(board[prev_x][prev_y]==AGNT_POS)
                            {
                                  board[prev_x][prev_y]=BLOCK;
                            }
                            board[ret_x][ret_y]=AGNT_POS;
                      }
                      //4b. alliws eksetazontai ta paidia tou
                      else
                      {
                            put_children(ret_x,ret_y,search_holes);
                            closure[clos_ind].x=ret_x;closure[clos_ind].y=ret_y;
                            clos_ind++;
                      }
                      /*prosomoiwsh kinhshs*/
                      prev_x=ret_x;prev_y=ret_y;
                      //An do8hke lysh adeiazei h closure
                      if((ret_x==final_x)&&(ret_y==final_y)){init_closure();return;}
                }
          }
    }
```

queue.h

```
 /*
```

```
Copyright (c) 2005-2008, Simon Howard

Permission to use, copy, modify, and/or distribute this software
for any purpose with or without fee is hereby granted, provided
that the above copyright notice and this permission notice appear
in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL
WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE
AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR
CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM
LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT,
NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN
CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

 */

/**
 * @file queue.h
 *
 * @brief Double-ended queue.
 *
 * A double ended queue stores a list of values in order.  New values
 * can be added and removed from either end of the queue.
 *
 * To create a new queue, use @ref queue_new.  To destroy a queue, use
 * @ref queue_free.
 *
 * To add values to a queue, use @ref queue_push_head and
 * @ref queue_push_tail.
```

```
 *
 * To read values from the ends of a queue, use @ref queue_pop_head
 * and @ref queue_pop_tail.  To examine the ends without removing values
 * from the queue, use @ref queue_peek_head and @ref queue_peek_tail.
 *
 */

#ifndef ALGORITHM_QUEUE_H
#define ALGORITHM_QUEUE_H

#ifdef __cplusplus
extern "C" {
#endif

/**
 * A double-ended queue.
 */

typedef struct _Queue Queue;

/**
 * A value stored in a @ref Queue.
 */

typedef void *QueueValue;

/**
 * A null @ref QueueValue.
 */

#define QUEUE_NULL ((void *) 0)
```

```
/**
 * Create a new double-ended queue.
 *
 * @return           A new queue, or NULL if it was not possible to allocate
 *                   the memory.
 */

Queue *queue_new(void);

/**
 * Destroy a queue.
 *
 * @param queue      The queue to destroy.
 */

void queue_free(Queue *queue);

/**
 * Add a value to the head of a queue.
 *
 * @param queue      The queue.
 * @param data       The value to add.
 * @return           Non-zero if the value was added successfully, or zero
 *                   if it was not possible to allocate the memory for the
 *                   new entry.
 */

int queue_push_head(Queue *queue, QueueValue data);

/**
 * Remove a value from the head of a queue.
 *
```

```
 * @param queue      The queue.
 * @return           Value that was at the head of the queue, or
 *                   @ref QUEUE_NULL if the queue is empty.
 */

QueueValue queue_pop_head(Queue *queue);

/**
 * Read value from the head of a queue, without removing it from
 * the queue.
 *
 * @param queue      The queue.
 * @return           Value at the head of the queue, or @ref QUEUE_NULL if the
 *                   queue is empty.
 */

QueueValue queue_peek_head(Queue *queue);

/**
 * Add a value to the tail of a queue.
 *
 * @param queue      The queue.
 * @param data       The value to add.
 * @return           Non-zero if the value was added successfully, or zero
 *                   if it was not possible to allocate the memory for the
 *                   new entry.
 */

int queue_push_tail(Queue *queue, QueueValue data);

/**
 * Remove a value from the tail of a queue.
```

```
 *
 * @param queue      The queue.
 * @return           Value that was at the head of the queue, or
 *                   @ref QUEUE_NULL if the queue is empty.
 */

QueueValue queue_pop_tail(Queue *queue);

/**
 * Read a value from the tail of a queue, without removing it from
 * the queue.
 *
 * @param queue      The queue.
 * @return           Value at the tail of the queue, or QUEUE_NULL if the
 *                   queue is empty.
 */

QueueValue queue_peek_tail(Queue *queue);

/**
 * Query if any values are currently in a queue.
 *
 * @param queue      The queue.
 * @return           Zero if the queue is not empty, non-zero if the queue
 *                   is empty.
 */

int queue_is_empty(Queue *queue);

int howmany(Queue *queue);
#ifdef __cplusplus
}
```

```
#endif

#endif /* #ifndef ALGORITHM_QUEUE_H */
```

queue.c

```
/*

Copyright (c) 2005-2008, Simon Howard

Permission to use, copy, modify, and/or distribute this software
for any purpose with or without fee is hereby granted, provided
that the above copyright notice and this permission notice appear
in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL
WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE
AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR
CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM
LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT,
NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN
CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

 */

#include <stdlib.h>

#include "queue.h"

/* malloc() / free() testing */
```

```
#ifdef ALLOC_TESTING
#include "alloc-testing.h"
#endif

/* A double-ended queue */

typedef struct _QueueEntry QueueEntry;

struct _QueueEntry {
   QueueValue data;
   QueueEntry *prev;
   QueueEntry *next;
};

struct _Queue {
   QueueEntry *head;
   QueueEntry *tail;
};

int nofq=0;

Queue *queue_new(void)
{
   Queue *queue;

   queue = (Queue *) malloc(sizeof(Queue));

   if (queue == NULL) {
         return NULL;
   }
```

```
    queue->head = NULL;
    queue->tail = NULL;

    return queue;
}

void queue_free(Queue *queue)
{
    /* Empty the queue */

    while (!queue_is_empty(queue)) {
          nofq--;queue_pop_head(queue);
    }

    /* Free back the queue */

    free(queue);
}

int queue_push_head(Queue *queue, QueueValue data)
{
    QueueEntry *new_entry;

    /* Create the new entry and fill in the fields in the structure */

    new_entry = malloc(sizeof(QueueEntry));

    if (new_entry == NULL) {
          return 0;
    }

    new_entry->data = data;
```

```
    new_entry->prev = NULL;
    new_entry->next = queue->head;

    /* Insert into the queue */

    if (queue->head == NULL) {

          /* If the queue was previously empty, both the head and tail must
           * be pointed at the new entry */

          queue->head = new_entry;
          queue->tail = new_entry;

    } else {

          /* First entry in the list must have prev pointed back to this
           * new entry */

          queue->head->prev = new_entry;

          /* Only the head must be pointed at the new entry */

          queue->head = new_entry;
    }
    nofq++;
    return 1;
}

QueueValue queue_pop_head(Queue *queue)
{
    QueueEntry *entry;
    QueueValue result;
```

```
    /* Check the queue is not empty */

    if (queue_is_empty(queue)) {
          return QUEUE_NULL;
    }

    /* Unlink the first entry from the head of the queue */

    entry = queue->head;
    result = entry->data;
    queue->head = entry->next;

    if (queue->head == NULL) {

          /* If doing this has unlinked the last entry in the queue, set
           * tail to NULL as well. */

          queue->tail = NULL;
    } else {

          /* The new first in the queue has no previous entry */

          queue->head->prev = NULL;
    }

    /* Free back the queue entry structure */
    nofq--;
    free(entry);

    return result;
}
```

```
QueueValue queue_peek_head(Queue *queue)
{
   if (queue_is_empty(queue)) {
         return QUEUE_NULL;
   } else {
         return queue->head->data;
   }
   nofq--;
}

int queue_push_tail(Queue *queue, QueueValue data)
{
   QueueEntry *new_entry;

   /* Create the new entry and fill in the fields in the structure */

   new_entry = malloc(sizeof(QueueEntry));

   if (new_entry == NULL) {
         return 0;
   }

   new_entry->data = data;
   new_entry->prev = queue->tail;
   new_entry->next = NULL;

   /* Insert into the queue tail */

   if (queue->tail == NULL) {

         /* If the queue was previously empty, both the head and tail must
```

```
         * be pointed at the new entry */

        queue->head = new_entry;
        queue->tail = new_entry;

    } else {

        /* The current entry at the tail must have next pointed to this
         * new entry */

        queue->tail->next = new_entry;

        /* Only the tail must be pointed at the new entry */

        queue->tail = new_entry;
    }
    nofq++;
    return 1;
}

QueueValue queue_pop_tail(Queue *queue)
{
    QueueEntry *entry;
    QueueValue result;

    /* Check the queue is not empty */

    if (queue_is_empty(queue)) {
        return QUEUE_NULL;
    }

    /* Unlink the first entry from the tail of the queue */
```

```
    entry = queue->tail;
    queue->tail = entry->prev;
    result = entry->data;

    if (queue->tail == NULL) {

          /* If doing this has unlinked the last entry in the queue, set
           * head to NULL as well. */

          queue->head = NULL;

    } else {

          /* The new entry at the tail has no next entry. */

          queue->tail->next = NULL;
    }

    /* Free back the queue entry structure */

    free(entry);
    nofq--;
    return result;
}

QueueValue queue_peek_tail(Queue *queue)
{
    if (queue_is_empty(queue)) {
          return QUEUE_NULL;
    } else {
          return queue->tail->data;
```

```
   }
}

int queue_is_empty(Queue *queue)
{
   return queue->head == NULL;
}

int howmany(Queue *queue)
{return nofq;}
```

tileworld.h

```
#include <stdio.h>
#include <stdlib.h>
#include "queue.h"
#include <pthread.h>
#define M 15
#define N 15
#define AGENTS 9  //ari8mos praktorwn
#define TW_ACTIONS 8 //tyxaies energies allaghs kosmou
#define BLOCK 0
#define BRICK 1
#define OBSTACLE 2
#define HOLE 3
#define FILLED 4
#define AGNT_POS 8

/*
paradoxes:
```

```
a)Statikos kosmos.
b)mh-epanametakinhsima touvla

   -     -     -

Symvolizoume me :

0 : "diadromo"
1 : touvla
2 : empodia
3 : kena
4: gemismenh trypa
8 : praktora

d)O praktoras arxika DEN mpainei se 8esh me touvlo
   -     -     -

O (prwtos) praktoras arxizei sth 8esh (2,1).
*/

// Struct gia epistrofh apo tis synarthseis pou xreiazetai twn metavlhtwn 8eshs
struct coord{
int x;
int y;
}dims;

struct id_coord{
int x;
int y;
int id;
};
```

```
int board[M][N];

void init_board(); /*arxikopoihsh pinaka*/


int count_data(int dt); /*epistrefei ton ekastote ari8mo twn kenwn*/


int init_holes;


void *randomizer();  /*synarthsh tyxaias allaghs tou kosmou*/


double euclidean_dist(int x,int x_brick,int y,int y_brick);


struct coord return_data(int data);
```

tileworld.c

```
#include <stdio.h>
#include <stdlib.h>
#include "queue.h"
#include "tileworld.h"
#include <pthread.h>


int init_holes=-1;

int count_data(int type)
{
    int i,j;
    int count=0;
    for(i=0;i<M;i++)
    {
          for(j=0;j<N;j++)
          { if(board[i][j]==type){count++;} }
```

```
    }
    if(type==HOLE&&init_holes<0){init_holes=count;}
    return count;
}


/*arxikopoihsh pinaka*/
void init_board()
{
    int i,j;
    for(i=0;i<M;i++)
    {
          for(j=0;j<N;j++)
          {board[i][j]=0;}
    }

    /*touvla*/
    board[11][14]=1;
    board[14][10]=1;
    board[5][3]=1;
    board[4][1]=1;
    board[7][3]=1;
    board[7][11]=1;
    board[11][7]=1;
    board[3][3]=1;
    board[4][2]=1;
    /*board[6][6]=1;
    board[1][1]=1;
    board[7][22]=1;
    board[11][5]=1;
    board[14][28]=1;
    board[3][25]=1;
    board[11][3]=1;
    board[12][18]=1;*/



    /*kena*/
    board[12][21]=3;
    board[1][3]=3;
    board[3][5]=3;
    board[5][9]=3;
    board[7][13]=3;
    board[13][14]=3;
    board[2][3]=3;
    board[9][5]=3;
    board[4][0]=3;
    /*board[13][27]=3;
    board[3][1]=3;
    board[3][18]=3;
    board[5][19]=3;
    board[7][23]=3;
    board[13][22]=3;
    board[2][23]=3;
    board[4][28]=3;
    board[14][13]=3;
    board[13][15]=3;
    board[2][2]=3; */

    /*empodia*/
    board[5][8]=2;
```

```
     board[11][9]=2;
     board[12][6]=2;
     board[6][1]=2;
     board[14][17]=2;
     board[10][8]=2;
     board[1][6]=2;
     board[8][11]=2;
     board[7][4]=2;
     board[14][8]=2;
     board[20][11]=2;
     /*board[8][12]=2;
     board[14][15]=2;
     board[11][23]=2;
     board[12][28]=2;
     board[9][15]=2;
     board[14][25]=2;
     board[13][12]=2;*/

     init_holes=count_data(HOLE);
}

//pthread_mutex_t mt=PTHREAD_MUTEX_INITIALIZER;


/*
     energeies tyxaiou kosmou

0    tipota
1    emfanish touvlou
2    eksafanish touvlou
3    emfanish trypas
4    eksafanish trypas
5    emfanish empodiou
6    eksafanish empodiou

oi emfaniseis/eksfaniseis twn stoixeiwn ginontai sto prwto dia8esimo tetragwno

*/

/*euresh prwtou dia8esimou touvlou,epistrefei tis syntetagmenes tou*/

void show_brick(int x,int y)
{printf("[ TW ]:I appear brick at (%d,%d)\n",x,y);board[x][y]=BRICK;}

void appear_block(int x,int y)
{printf("[ TW ]:I disappear brick at (%d,%d)\n",x,y);board[x][y]=BLOCK;}

void disappear_hole(int x,int y)
{printf("[ TW ]:I disappear hole at (%d,%d)\n",x,y);board[x][y]=BLOCK;}

void show_hole(int x,int y)
{printf("[ TW ]:I appear hole at (%d,%d)\n",x,y);board[x][y]=HOLE;}

void show_obstacle(int x,int y)
{printf("[ TW ]:I appear obstacle at (%d,%d)\n",x,y);board[x][y]=OBSTACLE;}

void disappear_obstacle(int x,int y)
{printf("[ TW ]:I disappear obstacle at (%d,%d)\n",x,y);board[x][y]=BLOCK;}


struct coord return_data(int data)
```

```
{
     struct coord dims_br;
     dims_br.x=-5;dims_br.y=-5;
     int i,j;
     for(i=0;i<M;i++)
     {
           for(j=0;j<N;j++)
           {
                 if(board[i][j]==data)
                 {
                       dims_br.x=i;
                       dims_br.y=j;
                       return dims_br;

                 }
           }
     }
     return dims_br;
}

void *randomizer()
{
     int action,sleep_time,rand_cnt;
     struct coord found;
     int i,j;

     for(rand_cnt=0; rand_cnt < TW_ACTIONS; rand_cnt++)
     {
           /*nea apofash tou tileworld se xrono apo 0 mexri 3 deuterolepta*/
           sleep_time=random()%4;

           /*epilogh energeias*/
           action=random()%7;
           if(action==1)
           {
                 found=return_data(BLOCK);
                 if(board[found.x][found.y]==BLOCK)
                 {show_brick(found.x,found.y);}
           }

           if(action==2)
           {
                 found = return_data(BRICK);
                 if(board[found.x][found.y]==BRICK)
                 {appear_block(found.x,found.y);}
           }

           if(action==3)
           {
                 found = return_data(BLOCK);
                 if(board[found.x][found.y]==BLOCK)
                 {
                       show_hole(found.x,found.y);
                       if(count_data(BRICK)>0){init_holes++;}
                 }
           }

           if(action==4)
           {
                 found = return_data(HOLE);
                 if(board[found.x][found.y]==HOLE)
```

```
                {
                        disappear_hole(found.x,found.y);
                        if(count_data(BRICK)>0){init_holes--;}
                }
        }

        if(action==5)
        {
                found = return_data(BLOCK);
                if(board[found.x][found.y]==BLOCK)
                {
                        show_obstacle(found.x,found.y);
                }
        }

        if(action==6)
        {

                        found = return_data(OBSTACLE);
                        if(board[found.x][found.y]==OBSTACLE)
                        {
                                disappear_obstacle(found.x,found.y);
                        }
        }

        sleep(sleep_time);
    }
}

}
```

tileworld_tester.c

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <pthread.h>
#include "agent.h"

int main()
{
    struct id_coord *inp[AGENTS];
    pthread_t threads[AGENTS+1];
    int i;

    init_board();

    for(i=0;i<AGENTS;i++)
    {inp[i]= (struct id_coord *)malloc(sizeof(struct coord));}

    /*polypraktoriko systhma: o a' ksekinaei apo to (2,1) kai oi ypoloipoi ana 1 block
diagwniws*/
    inp[0]->x=2;inp[0]->y=1;inp[0]->id=0;
    for(i=1;i<AGENTS;i++)
    {
        inp[i]->x=inp[i-1]->x+1;
        inp[i]->y=inp[i-1]->y+1;inp[i]->id=inp[i-1]->id+1;
    }
```

```
	pthread_create(&threads[AGENTS],NULL,(void *)randomizer,NULL);
	sleep(2);
	for(i=0;i<AGENTS;i++)
	{
		pthread_create(&threads[i],NULL,(void *)agent,inp[i]);
	}

	pthread_join(threads[AGENTS],NULL);
	sleep(4);
	for(i=0;i<AGENTS;i++)
	{
			pthread_join(threads[i],NULL);
	}


	for(i=0;i<AGENTS;i++)
	{
		free(inp[i]);
	}
	printf("\n\t\t* * *\n\n");

}
```

Για μεταγλώττιση του προγράμματος χρησιμοποιείται η εντολή:

gcc tileworld.c queue.c mybfs.c agent.c tileworld_tester.c -o ag -lm -lpthread

και το πρόγραμμα τρέχει με :

./ag

Συμπεράσματα εκτελέσεων προγράμματος

Ύστερα από αρκετές επανεκτελέσεις ορισμένων εκδοχών με αλλαγή συγκεκριμένων παραμέτρων, συνήχθησαν συμπεράσματα ως προς τη συμπεριφορά του προγράμματος και κατά πόσον επηρεάζεται από αυτές. Κάθε εκτέλεση του προγράμματος έγινε με ανακατεύθυνση εισόδου σε ένα .txt αρχείο. Με (AG-i) , $0 \leq I < AGENTS$, συμβολίζεται η ενέργεια του πράκτορα με αριθμό i, με [TW] η ενέργεια του κόσμου. Όπου υπήρχαν μεγάλες αποκλίσεις στις επανεκτελέσεις του προγράμματος, έγιναν περισσότερες από το μέσο όρο (περίπου 3).

Εκδοχή 1.1:

ΠΑΡΑΜΕΤΡΟΣ	ΤΙΜΗ
M	15
N	15
Τούβλα	1
Τρύπες	3
Εμπόδια	11
Ενέργειες TW (μέγιστο)	8
Πράκτορες	1

ΕΞΟΔΟΣ	ΤΙΜΗ
Μέσος χρόνος διεκπεραίωσης	2.3
Διασπορά χρόνου διεκπεραίωσης	0.33
Μέση βαθμολογία πράκτορα	1
Διασπορά βαθμολογίας	0
Τυπική απόκλιση βαθμολογίας	0
Άθροισμα βαθμολογιών	1
Ενεργοί πράκτορες (βαθμ. > 0)	100%
Ποσοστό (μη αδρανών) ενεργειών TW	100%

Εκτέλεση 1.1α

```
[ TW ]:I disappear hole at (1,3)
(AG-0) : Brick (11,14) at (3,5) ,1/1 totally


      ---(AG-0) time :2.00,   Score:1.000

[ TW ]:I appear brick at (0,0)
[ TW ]:I appear hole at (0,1)
[ TW ]:I appear brick at (0,2)
[ TW ]:I disappear brick at (0,0)
[ TW ]:I disappear obstacle at (1,6)
[ TW ]:I disappear obstacle at (4,0)
[ TW ]:I appear hole at (0,0)

            * * *
```

Εκτέλεση 1.1β

```
[ TW ]:I disappear hole at (1,3)
(AG-0) : Brick (11,14) at (3,5) ,1/1 totally


      ---(AG-0) time :2.00,   Score:1.000
```

```
[ TW ]:I appear brick at (0,0)
[ TW ]:I appear hole at (0,1)
[ TW ]:I appear brick at (0,2)
[ TW ]:I disappear brick at (0,0)
[ TW ]:I disappear obstacle at (1,6)
[ TW ]:I disappear obstacle at (4,0)
[ TW ]:I appear hole at (0,0)

            * * *
```

Εκτέλεση 1.1γ

```
[ TW ]:I disappear hole at (1,3)
(AG-0) : Brick (11,14) at (3,5) ,1/1 totally


      ---(AG-0) time :3.00,   Score:1.000

[ TW ]:I appear brick at (0,0)
[ TW ]:I appear hole at (0,1)
[ TW ]:I appear brick at (0,2)
[ TW ]:I disappear brick at (0,0)
[ TW ]:I disappear obstacle at (1,6)
[ TW ]:I disappear obstacle at (4,0)
[ TW ]:I appear hole at (0,0)

                  * * *
```

Συμπεράσματα: Παρατηρείται ότι στο μονοπρακτορικό σύστημα με μία τρύπα, ένα τούβλο, και μέτριο αριθμό εμποδίων και ενεργειών ο πράκτορας και ο κόσμος ολοκληρώνουν επιτυχώς τις εργασίες τους

Εκδοχή 1.2:

ΠΑΡΑΜΕΤΡΟΣ	ΤΙΜΗ
M	15
N	30
Τούβλα	18
Τρύπες	18
Εμπόδια	15
Ενέργειες TW (μέγιστο)	5
Πράκτορες	1

ΕΞΟΔΟΣ	**ΤΙΜΗ**
Μέσος χρόνος διεκπεραίωσης	2.5
Διασπορά χρόνου διεκπεραίωσης	0.33
Μέση βαθμολογία πράκτορα	1
Διασπορά βαθμολογίας	0
Τυπική απόκλιση βαθμολογίας	0
Άθροισμα βαθμολογιών	1
Ενεργοί πράκτορες (βαθμ. > 0)	100%
Ποσοστό (μη αδρανών) ενεργειών TW	100%

Εκτέλεση 1.2α

```
[ TW ]:I disappear hole at (1,3)
(AG-0) : Brick (1,1) at (2,3) ,1/17 totally

(AG-0) : Brick (2,2) at (2,23) ,2/17 totally

(AG-0) : Brick (3,3) at (3,1) ,3/17 totally

(AG-0) : Brick (3,25) at (3,5) ,4/17 totally

(AG-0) : Brick (4,1) at (3,18) ,5/17 totally

(AG-0) : Brick (4,2) at (4,0) ,6/17 totally

(AG-0) : Brick (5,3) at (4,28) ,7/17 totally

(AG-0) : Brick (6,6) at (5,9) ,8/17 totally

(AG-0) : Brick (7,3) at (5,19) ,9/17 totally

(AG-0) : Brick (7,11) at (7,13) ,10/17 totally

(AG-0) : Brick (7,22) at (7,23) ,11/17 totally

(AG-0) : Brick (11,3) at (9,5) ,12/17 totally

(AG-0) : Brick (11,5) at (9,15) ,13/17 totally

(AG-0) : Brick (11,7) at (13,12) ,14/17 totally

(AG-0) : Brick (11,14) at (13,22) ,15/17 totally

(AG-0) : Brick (12,18) at (14,13) ,16/17 totally

(AG-0) : Brick (13,14) at (14,25) ,17/17 totally


      ---(AG-0) time :2.00,   Score:1.000

[ TW ]:I appear hole at (0,0)
[ TW ]:I appear obstacle at (0,2)
[ TW ]:I disappear brick at (13,27)
```

```
[ TW ]:I appear obstacle at (0,3)
```

* * *

Εκτέλεση 1.2β

```
[ TW ]:I disappear hole at (1,3)
(AG-0) : Brick (1,1) at (2,3) ,1/17 totally

(AG-0) : Brick (2,2) at (2,23) ,2/17 totally

(AG-0) : Brick (3,3) at (3,1) ,3/17 totally

(AG-0) : Brick (3,25) at (3,5) ,4/17 totally

(AG-0) : Brick (4,1) at (3,18) ,5/17 totally

(AG-0) : Brick (4,2) at (4,0) ,6/17 totally

(AG-0) : Brick (5,3) at (4,28) ,7/17 totally

(AG-0) : Brick (6,6) at (5,9) ,8/17 totally

(AG-0) : Brick (7,3) at (5,19) ,9/17 totally

(AG-0) : Brick (7,11) at (7,13) ,10/17 totally

(AG-0) : Brick (7,22) at (7,23) ,11/17 totally

(AG-0) : Brick (11,3) at (9,5) ,12/17 totally

(AG-0) : Brick (11,5) at (9,15) ,13/17 totally

(AG-0) : Brick (11,7) at (13,12) ,14/17 totally

(AG-0) : Brick (11,14) at (13,22) ,15/17 totally

(AG-0) : Brick (12,18) at (14,13) ,16/17 totally

(AG-0) : Brick (13,14) at (14,25) ,17/17 totally


      ---(AG-0) time :3.00,   Score:1.000

[ TW ]:I appear hole at (0,0)
[ TW ]:I appear obstacle at (0,2)
[ TW ]:I disappear brick at (13,27)
[ TW ]:I appear obstacle at (0,3)
```

* * *

Εκτέλεση 1.2γ

```
[ TW ]:I disappear hole at (1,3)
(AG-0) : Brick (1,1) at (2,3) ,1/17 totally

(AG-0) : Brick (2,2) at (2,23) ,2/17 totally

(AG-0) : Brick (3,3) at (3,1) ,3/17 totally
```

```
(AG-0) : Brick (3,25) at (3,5) ,4/17 totally

(AG-0) : Brick (4,1) at (3,18) ,5/17 totally

(AG-0) : Brick (4,2) at (4,0) ,6/17 totally

(AG-0) : Brick (5,3) at (4,28) ,7/17 totally

(AG-0) : Brick (6,6) at (5,9) ,8/17 totally

(AG-0) : Brick (7,3) at (5,19) ,9/17 totally

(AG-0) : Brick (7,11) at (7,13) ,10/17 totally

(AG-0) : Brick (7,22) at (7,23) ,11/17 totally

(AG-0) : Brick (11,3) at (9,5) ,12/17 totally

(AG-0) : Brick (11,5) at (9,15) ,13/17 totally

(AG-0) : Brick (11,7) at (13,12) ,14/17 totally

(AG-0) : Brick (11,14) at (13,22) ,15/17 totally

(AG-0) : Brick (12,18) at (14,13) ,16/17 totally

(AG-0) : Brick (13,14) at (14,25) ,17/17 totally


      ---(AG-0) time :3.00,   Score:1.000

[ TW ]:I appear hole at (0,0)
[ TW ]:I appear obstacle at (0,2)
[ TW ]:I disappear brick at (13,27)
[ TW ]:I appear obstacle at (0,3)

                  * * *
```

Εκτέλεση 1.2δ

```
[ TW ]:I disappear hole at (1,3)
(AG-0) : Brick (1,1) at (2,3) ,1/17 totally

(AG-0) : Brick (2,2) at (2,23) ,2/17 totally

(AG-0) : Brick (3,3) at (3,1) ,3/17 totally

(AG-0) : Brick (3,25) at (3,5) ,4/17 totally

(AG-0) : Brick (4,1) at (3,18) ,5/17 totally

(AG-0) : Brick (4,2) at (4,0) ,6/17 totally

(AG-0) : Brick (5,3) at (4,28) ,7/17 totally

(AG-0) : Brick (6,6) at (5,9) ,8/17 totally
```

```
(AG-0) : Brick (7,3) at (5,19) ,9/17 totally

(AG-0) : Brick (7,11) at (7,13) ,10/17 totally

(AG-0) : Brick (7,22) at (7,23) ,11/17 totally

(AG-0) : Brick (11,3) at (9,5) ,12/17 totally

(AG-0) : Brick (11,5) at (9,15) ,13/17 totally

(AG-0) : Brick (11,7) at (13,12) ,14/17 totally

(AG-0) : Brick (11,14) at (13,22) ,15/17 totally

(AG-0) : Brick (12,18) at (14,13) ,16/17 totally

(AG-0) : Brick (13,14) at (14,25) ,17/17 totally


      ---(AG-0) time :2.00,   Score:1.000

[ TW ]:I appear hole at (0,0)
[ TW ]:I appear obstacle at (0,2)
[ TW ]:I disappear brick at (13,27)
[ TW ]:I appear obstacle at (0,3)

            * * *
```

Συμπεράσματα: Όπως και στην προηγούμενη εκδοχή, διεκπεραιώνονται οι εργασίες κόσμου πράκτορα με ελαφρώς αυξημένο μέσο όρο διεκπεραίωσης, πιθανότατα λόγω αυξημένου φόρτου (περισσότερα τούβλα – τρύπες). Οι αδρανείς ενέργειες, που είναι ζήτημα εκτέλεσης της συνάρτησης τυχαιοποίησης *(* `random()%7 == 0` *)* είναι μηδέν. Επιπλέον το ένα τούβλο που απομένει μετά το γέμισμα όλων των τρυπών ο πράκτορας δεν το λαμβάνει υπ όψιν και τυχαία ο κόσμος προλαβαίνει και το εξαφανίζει.

Εκδοχή 2.1:

ΠΑΡΑΜΕΤΡΟΣ	ΤΙΜΗ
M	15
N	15
Τούβλα	4
Τρύπες	4
Εμπόδια	11
Ενέργειες TW (μέγιστο)	5
Πράκτορες	2

ΕΞΟΔΟΣ	**ΤΙΜΗ**
Μέσος χρόνος διεκπεραίωσης	5.17
Διασπορά χρόνου διεκπεραίωσης	0.566
Μέση βαθμολογία πράκτορα	0.5
Διασπορά βαθμολογίας	0.236
Τυπική απόκλιση βαθμολογίας	0.056
Ενεργοί πράκτορες (βαθμ. > 0)	100%
Ποσοστό (μη αδρανών) ενεργειών TW	80%

Εκτέλεση 2.1α

```
[ TW ]:I disappear hole at (0,1)
(AG-1) : Brick (0,5) at (1,3) ,1/3 totally

(AG-0) : Brick (1,0) at (2,3) ,1/3 totally

(AG-1) : Brick (4,1) at (3,5) ,2/3 totally

(AG-0) : Brick (5,4) destroyed

      ---(AG-1) time :5.00,   Score:0.667---


      ---(AG-0) time :4.00,   Score:0.333---

[ TW ]:I appear hole at (0,1)
[ TW ]:I appear obstacle at (0,2)
[ TW ]:I disappear hole at (0,1)

            * * *
```

Εκτέλεση 2.1β

```
[ TW ]:I disappear hole at (0,1)
(AG-1) : Brick (0,5) at (1,3) ,1/3 totally

(AG-0) : Brick (1,0) at (2,3) ,1/3 totally

(AG-1) : Brick (4,1) at (3,5) ,2/3 totally

(AG-0) : Brick (5,4) destroyed

      ---(AG-1) time :5.00,   Score:0.667---


      ---(AG-0) time :5.00,   Score:0.333---
```

```
[ TW ]:I appear hole at (0,1)
[ TW ]:I appear obstacle at (0,2)
[ TW ]:I disappear hole at (0,1)

            * * *
```

Εκτέλεση 2.1γ

```
[ TW ]:I disappear hole at (0,1)
(AG-1) : Brick (0,5) at (1,3) ,1/3 totally

(AG-0) : Brick (1,0) at (2,3) ,1/3 totally

(AG-1) : Brick (4,1) at (3,5) ,2/3 totally

(AG-0) : Brick (5,4) destroyed

      ---(AG-0) time :6.00,   Score:0.333---


      ---(AG-1) time :6.00,   Score:0.667---

[ TW ]:I appear hole at (0,1)
[ TW ]:I appear obstacle at (0,2)
[ TW ]:I disappear hole at (0,1)

                  * * *
```

Συμπεράσματα: Ο μέσος χρόνος διεκπεραίωσης είναι περίπου διπλάσιος από το αντίστοιχο σενάριο ως προς το φόρτο και τα εμπόδια στο μονοπρακτορικό σύστημα, καθώς οι πράκτορες αυξήθηκαν μεν αλλά και το φόρτο τους διπλασιάστηκε. Η διασπορά και η τυπική απόκλιση βαθμολογίας μεγαλώνουν καθώς ο ένας πράκτορας κάθε φορά γεμίζει τις διπλάσιες τρύπες από τον άλλον. Τέλος ο κόσμος εμφανίζει 20% αδρανείς ενέργειες. Αξίζει να σημειωθεί και η σωστή αντίδραση του πράκτορα να καταστρέψει το τούβλο που δε χρειάζεται.

Εκδοχή 2.2:

ΠΑΡΑΜΕΤΡΟΣ	ΤΙΜΗ
M	12
N	12
Τούβλα	7
Τρύπες	6
Εμπόδια	17
Ενέργειες TW (μέγιστο)	6
Πράκτορες	2

ΕΞΟΔΟΣ	**ΤΙΜΗ**
Μέσος χρόνος διεκπεραίωσης	5.5
Διασπορά χρόνου διεκπεραίωσης	0.7
Μέση βαθμολογία πράκτορα	0.5
Διασπορά βαθμολογίας	0.18
Τυπική απόκλιση βαθμολογίας	0.424
Ενεργοί πράκτορες (βαθμ. > 0)	100%
Ποσοστό (μη αδρανών) ενεργειών TW	83%

Εκτέλεση 2.2α

```
[ TW ]:I disappear hole at (0,1)
(AG-1) : Brick (0,6) at (1,3) ,1/5 totally

(AG-0) : Brick (1,0) at (3,5) ,1/5 totally

(AG-1) : Brick (4,1) at (8,9) ,2/5 totally

(AG-1) : Brick (5,4) at (11,9) ,3/5 totally

(AG-1) : Brick (7,3) at (11,11) ,4/5 totally

(AG-0) : Brick (7,11) destroyed

      ---(AG-1) time :5.00,   Score:0.800


      ---(AG-0) time :4.00,   Score:0.200

[ TW ]:I appear hole at (0,1)
[ TW ]:I appear obstacle at (0,2)
[ TW ]:I disappear hole at (0,1)
[ TW ]:I appear brick at (0,1)

            * * *
```

Εκτέλεση 2.2β

```
[ TW ]:I disappear hole at (0,1)
(AG-1) : Brick (0,6) at (1,3) ,1/5 totally

(AG-0) : Brick (1,0) at (3,5) ,1/5 totally

(AG-1) : Brick (4,1) at (8,9) ,2/5 totally

(AG-1) : Brick (5,4) at (11,9) ,3/5 totally

(AG-1) : Brick (7,3) at (11,11) ,4/5 totally

(AG-0) : Brick (7,11) destroyed
```

```
      ---(AG-1) time :6.00,   Score:0.800


      ---(AG-0) time :6.00,   Score:0.200

[ TW ]:I appear hole at (0,1)
[ TW ]:I appear obstacle at (0,2)
[ TW ]:I disappear hole at (0,1)
[ TW ]:I appear brick at (0,1)

            * * *
```

Εκτέλεση 2.2γ

```
[ TW ]:I disappear hole at (0,1)
(AG-1) : Brick (0,6) at (1,3) ,1/5 totally

(AG-0) : Brick (1,0) at (3,5) ,1/5 totally

(AG-1) : Brick (4,1) at (8,9) ,2/5 totally

(AG-1) : Brick (5,4) at (11,9) ,3/5 totally

(AG-1) : Brick (7,3) at (11,11) ,4/5 totally

(AG-0) : Brick (7,11) destroyed

      ---(AG-0) time :6.00,   Score:0.200


      ---(AG-1) time :6.00,   Score:0.800

[ TW ]:I appear hole at (0,1)
[ TW ]:I appear obstacle at (0,2)
[ TW ]:I disappear hole at (0,1)
[ TW ]:I appear brick at (0,1)
```

Συμπεράσματα: Σ' αυτή την εκδοχή παρατηρείται ότι η διασπορά του χρόνου διεκπεραίωσης έχει αυξηθεί που εν μέρει δικαιολογείται από το μεγαλύτερο φόρτο των πρακτόρων (αύξηση τρυπών και τούβλων) και από τις μεγαλύτερες αποστάσεις που καλύπτει ο πράκτορας 0. Η μέση βαθμολογία παραμένει στο 0.5 αλλά αυξήθηκε η διασπορά και η τυπική της απόκλιση αφού παρατηρείται ότι ο ένας πράκτορας επιτελεί τετραπλάσιο έργο από τον άλλο. Ο μέσος χρόνος διεκπεραίωσης σε σχέση με το αντίστοιχου φόρτου μονοπρακτορικό σύστημα υπερδιπλασιάστηκε, αφού αυξήθηκε αντίστοιχα και ο αριθμός των εμποδίων.

Εκδοχή 2.3:

ΠΑΡΑΜΕΤΡΟΣ	ΤΙΜΗ
M	30
N	33
Τούβλα	7
Τρύπες	6
Εμπόδια	17
Ενέργειες TW (μέγιστο)	10
Πράκτορες	2

ΕΞΟΔΟΣ	ΤΙΜΗ
Μέσος χρόνος διεκπεραίωσης	5.166
Διασπορά χρόνου διεκπεραίωσης	0.966
Μέση βαθμολογία πράκτορα	0.333
Διασπορά βαθμολογίας	0.033
Τυπική απόκλιση βαθμολογίας	0.182
Άθροισμα βαθμολογιών	0.667
Ενεργοί πράκτορες (βαθμ. > 0)	100%
Ποσοστό (μη αδρανών) ενεργειών TW	100%

Εκτέλεση 2.3α

```
[ TW ]:I disappear brick at (4,1)
[ TW ]:I disappear hole at (1,3)
[ TW ]:I disappear brick at (5,3)
[ TW ]:I appear hole at (0,0)
(AG-1) : Brick (7,3) at (0,0) ,1/6 totally

(AG-1) : Brick (7,11) at (2,13) ,2/6 totally

(AG-1) : Brick (11,7) at (3,5) ,3/6 totally

(AG-0) : Brick (11,14) at (3,26) ,1/6 totally

[ TW ]:I disappear brick at (14,10)
[ TW ]:I appear obstacle at (0,0)

      ---(AG-1) time :6.00,   Score:0.500


      ---(AG-0) time :5.00,   Score:0.167

[ TW ]:I disappear hole at (2,13)
[ TW ]:I appear brick at (0,1)
[ TW ]:I disappear brick at (0,1)
[ TW ]:I appear obstacle at (0,1)
```

```
            * * *
```

Εκτέλεση 2.3β

```
[ TW ]:I disappear brick at (4,1)
[ TW ]:I disappear hole at (1,3)
[ TW ]:I disappear brick at (5,3)
[ TW ]:I appear hole at (0,0)
(AG-1) : Brick (7,3) at (0,0) ,1/6 totally

(AG-1) : Brick (7,11) at (2,13) ,2/6 totally

(AG-1) : Brick (11,7) at (3,5) ,3/6 totally

(AG-0) : Brick (11,14) at (3,26) ,1/6 totally

[ TW ]:I disappear brick at (14,10)
[ TW ]:I appear obstacle at (0,0)

      ---(AG-1) time :6.00,   Score:0.500


      ---(AG-0) time :4.00,   Score:0.167

[ TW ]:I disappear hole at (2,13)
[ TW ]:I appear brick at (0,1)
[ TW ]:I disappear brick at (0,1)
[ TW ]:I appear obstacle at (0,1)

            * * *
```

Εκτέλεση 2.3γ

```
[ TW ]:I disappear brick at (4,1)
[ TW ]:I disappear hole at (1,3)
[ TW ]:I disappear brick at (5,3)
[ TW ]:I appear hole at (0,0)
(AG-1) : Brick (7,3) at (0,0) ,1/6 totally

(AG-1) : Brick (7,11) at (2,13) ,2/6 totally

(AG-1) : Brick (11,7) at (3,5) ,3/6 totally

(AG-0) : Brick (11,14) at (3,26) ,1/6 totally

[ TW ]:I disappear brick at (14,10)
[ TW ]:I appear obstacle at (0,0)

      ---(AG-1) time :6.00,   Score:0.500


      ---(AG-0) time :4.00,   Score:0.167

[ TW ]:I disappear hole at (2,13)
[ TW ]:I appear brick at (0,1)
[ TW ]:I disappear brick at (0,1)
[ TW ]:I appear obstacle at (0,1)

            * * *
```

Συμπεράσματα: Το άθροισμα βαθμολογιών για πρώτη φορά και μετά την αύξηση των ενεργειών του κόσμου γίνεται μικρότερο της μονάδας. Επιπλέον ο κόσμος παρεμβάλλεται περισσότερο μεταξύ της εκτέλεσης των πρακτόρων. Ο μέσος χρόνος διεκπεραίωσης μικραίνει αρκετά, πιθανότατα λόγω αραιότερου και μεγαλύτερου κόσμου (και άρα περισσότερων εναλλακτικών στην κίνηση). Επίσης η διασπορά μειώθηκε που σημαίνει ότι οι πράκτορες είναι λιγότερο ανταγωνιστικοί.

Εκδοχή 3.1:

ΠΑΡΑΜΕΤΡΟΣ	ΤΙΜΗ
M	15
N	15
Τούβλα	8
Τρύπες	7
Εμπόδια	11
Ενέργειες TW (μέγιστο)	8
Πράκτορες	4

ΕΞΟΔΟΣ	ΤΙΜΗ
Μέσος χρόνος διεκπεραίωσης	6.5
Διασπορά χρόνου διεκπεραίωσης	2.09
Μέση βαθμολογία πράκτορα	0.25
Διασπορά βαθμολογίας	0.03
Τυπική απόκλιση βαθμολογίας	0.173
Ενεργοί πράκτορες (βαθμ. > 0)	75%
Ποσοστό (μη αδρανών) ενεργειών TW	62.5%

Εκτέλεση 3.1α

```
[ TW ]:I disappear hole at (1,3)
[ TW ]:I disappear brick at (3,3)
(AG-3) : Brick (4,1) at (2,3) ,1/6 totally

(AG-1) : Brick (5,3) at (3,5) ,1/6 totally

(AG-3) : Brick (7,3) at (5,9) ,2/6 totally

(AG-1) : Brick (7,11) at (7,13) ,2/6 totally

[ TW ]:I disappear hole at (13,6)
(AG-2) : Brick (11,7) at (13,14) ,1/5 totally

(AG-0) : Brick (11,14) destroyed

      ---(AG-2) time :5.00,   Score:0.200
```

```
      ---(AG-0) time :6.00,   Score:0.000

(AG-3) : Brick (14,10) destroyed

      ---(AG-3) time :6.00,   Score:0.400


      ---(AG-1) time :9.00,   Score:0.400

[ TW ]:I appear brick at (0,0)
[ TW ]:I disappear brick at (0,0)

                  * * *
```

Εκτέλεση 3.1β

```
[ TW ]:I disappear hole at (1,3)
[ TW ]:I disappear brick at (3,3)
(AG-0) : Brick (4,1) at (2,3) ,1/6 totally

(AG-2) : Brick (5,3) at (3,5) ,1/6 totally

(AG-0) : Brick (7,3) at (5,9) ,2/6 totally

(AG-2) : Brick (7,11) at (7,13) ,2/6 totally

[ TW ]:I disappear hole at (13,6)
(AG-1) : Brick (11,7) at (13,14) ,1/5 totally

(AG-3) : Brick (11,14) destroyed

      ---(AG-3) time :6.00,   Score:0.000


      ---(AG-1) time :5.00,   Score:0.200

(AG-0) : Brick (14,10) destroyed

      ---(AG-0) time :7.00,   Score:0.400


      ---(AG-2) time :9.00,   Score:0.400

[ TW ]:I appear brick at (0,0)
[ TW ]:I disappear brick at (0,0)

                  * * *
```

Εκτέλεση 3.1γ

```
[ TW ]:I disappear hole at (1,3)
[ TW ]:I disappear brick at (3,3)
(AG-2) : Brick (4,1) at (2,3) ,1/6 totally

(AG-1) : Brick (5,3) at (3,5) ,1/6 totally

(AG-2) : Brick (7,3) at (5,9) ,2/6 totally
```

```
(AG-1) : Brick (7,11) at (7,13) ,2/6 totally

[ TW ]:I disappear hole at (13,6)
(AG-3) : Brick (11,7) at (13,14) ,1/5 totally

(AG-0) : Brick (11,14) destroyed

      ---(AG-3) time :6.00,   Score:0.200


      ---(AG-0) time :5.00,   Score:0.000

(AG-2) : Brick (14,10) destroyed

      ---(AG-2) time :6.00,   Score:0.400


      ---(AG-1) time :8.00,   Score:0.400

[ TW ]:I appear brick at (0,0)
[ TW ]:I disappear brick at (0,0)

            * * *
```

Συμπεράσματα: Ο μέσος χρόνος διεκπεραίωσης και η διασπορά του αυξήθηκαν σε σχέση με το σύστημα δυο πρακτόρων, αφού οι συνολικές διαθέσιμες επιλογές για τον πράκτορα αυξάνονται. Παρόλα αυτά παρατηρείται σημαντικό κέρδος αφού είναι μικρότερος ο μέσος χρόνος απ τον διπλάσιο του συστήματος αντίστοιχου φόρτου δύο πρακτόρων και πολύ μικρότερος από το τετραπλάσιο μονοπρακτορικό. Η τυπική απόκλιση και διασπορά της βαθμολογίας μειώθηκαν αισθητά καθώς βλέπουμε τις τιμές να είναι πιο κοντά στο μέσο όρο αλλά και μεταξύ τους απ ο, τι πριν που ο ένας πράκτορας συγκέντρωνε την τετραπλάσια βαθμολογία του άλλου ως συνέπεια της μείωσης των ενεργών πρακτόρων. Τέλος η παρεμβολή του κόσμου ανάμεσα στους πράκτορες είναι μικρή και το άθροισμα βαθμολογιών 1, άρα μικρός και ο «ανταγωνισμός» τους.

Εκδοχή 3.2:

ΠΑΡΑΜΕΤΡΟΣ	ΤΙΜΗ
M	15
N	30
Τούβλα	7
Τρύπες	7
Εμπόδια	0
Ενέργειες TW (μέγιστο)	8
Πράκτορες	4

ΕΞΟΔΟΣ	**ΤΙΜΗ**
Μέσος χρόνος διεκπεραίωσης	6.583
Διασπορά χρόνου διεκπεραίωσης	2.083
Μέση βαθμολογία πράκτορα	0.25
Διασπορά βαθμολογίας	0.036
Τυπική απόκλιση βαθμολογίας	0.191
Ενεργοί πράκτορες (βαθμ. > 0)	75%
Ποσοστό (μη αδρανών) ενεργειών TW	87.5%

Εκτέλεση 3.2α

```
[ TW ]:I disappear hole at (1,3)
[ TW ]:I disappear brick at (4,1)
(AG-3) : Brick (5,3) at (2,3) ,1/6 totally

(AG-1) : Brick (7,3) at (3,5) ,1/6 totally

(AG-3) : Brick (7,11) at (5,9) ,2/6 totally

(AG-1) : Brick (11,7) at (7,13) ,2/6 totally

[ TW ]:I disappear hole at (12,21)
(AG-2) : Brick (11,14) at (13,14) ,1/5 totally

(AG-0) : Brick (14,10) destroyed

      ---(AG-0) time :6.00,   Score:0.000


      ---(AG-2) time :5.00,   Score:0.200


      ---(AG-3) time :6.00,   Score:0.400


      ---(AG-1) time :8.00,   Score:0.400

[ TW ]:I appear brick at (0,0)
[ TW ]:I disappear brick at (0,0)
[ TW ]:I appear obstacle at (0,0)
[ TW ]:I disappear obstacle at (0,0)

                    * * *
```

Εκτέλεση 3.2β

```
[ TW ]:I disappear hole at (1,3)
[ TW ]:I disappear brick at (4,1)
(AG-3) : Brick (5,3) at (2,3) ,1/6 totally

(AG-1) : Brick (7,3) at (3,5) ,1/6 totally

(AG-3) : Brick (7,11) at (5,9) ,2/6 totally

(AG-1) : Brick (11,7) at (7,13) ,2/6 totally
```

```
[ TW ]:I disappear hole at (12,21)
(AG-2) : Brick (11,14) at (13,14) ,1/5 totally

(AG-0) : Brick (14,10) destroyed

      ---(AG-0) time :6.00,   Score:0.000


      ---(AG-2) time :5.00,   Score:0.200


      ---(AG-3) time :7.00,   Score:0.400


      ---(AG-1) time :9.00,   Score:0.400

[ TW ]:I appear brick at (0,0)
[ TW ]:I disappear brick at (0,0)
[ TW ]:I appear obstacle at (0,0)
[ TW ]:I disappear obstacle at (0,0)
```

Εκτέλεση 3.2γ

```
[ TW ]:I disappear hole at (1,3)
[ TW ]:I disappear brick at (4,1)
(AG-3) : Brick (5,3) at (2,3) ,1/6 totally

(AG-1) : Brick (7,3) at (3,5) ,1/6 totally

(AG-3) : Brick (7,11) at (5,9) ,2/6 totally

(AG-1) : Brick (11,7) at (7,13) ,2/6 totally

[ TW ]:I disappear hole at (12,21)
(AG-2) : Brick (11,14) at (13,14) ,1/5 totally

(AG-0) : Brick (14,10) destroyed

      ---(AG-0) time :6.00,   Score:0.000


      ---(AG-2) time :5.00,   Score:0.200


      ---(AG-3) time :7.00,   Score:0.400


      ---(AG-1) time :9.00,   Score:0.400

[ TW ]:I appear brick at (0,0)
[ TW ]:I disappear brick at (0,0)
[ TW ]:I appear obstacle at (0,0)
[ TW ]:I disappear obstacle at (0,0)

            * * *
```

Συμπεράσματα: Ο κόσμος εν σχέση με πριν είναι πιο αραιός και μεγαλύτερος άρα αναμένουμε μείωση του μέσου χρόνου. Παρόλα αυτά μια ατυχής ενέργεια του πράκτορα 1 που διαλέγει μακρινά τούβλα και τρύπες ανεβάζει το μέσο όρο και τη διασπορά του.

Εκδοχή 3.3:

ΠΑΡΑΜΕΤΡΟΣ	ΤΙΜΗ
M	15
N	15
Τούβλα	16
Τρύπες	16
Εμπόδια	0
Ενέργειες TW (μέγιστο)	20
Πράκτορες	4

ΕΞΟΔΟΣ	ΤΙΜΗ
Μέσος χρόνος διεκπεραίωσης	10.416
Διασπορά χρόνου διεκπεραίωσης	0.992
Μέση βαθμολογία πράκτορα	0.25
Διασπορά βαθμολογίας	0.008
Τυπική απόκλιση βαθμολογίας	0.092
Ενεργοί πράκτορες (βαθμ. > 0)	100%
Ποσοστό (μη αδρανών) ενεργειών TW	90%

Εκτέλεση 3.3α

```
[ TW ]:I disappear hole at (1,3)
[ TW ]:I disappear brick at (1,1)
(AG-3) : Brick (3,3) at (2,3) ,1/15 totally

(AG-1) : Brick (3,7) at (2,13) ,1/15 totally

(AG-3) : Brick (4,1) at (3,1) ,2/15 totally

(AG-1) : Brick (4,2) at (3,5) ,2/15 totally

[ TW ]:I disappear hole at (3,6)
(AG-2) : Brick (5,3) at (4,0) ,1/14 totally

(AG-0) : Brick (6,6) at (5,9) ,1/14 totally

(AG-3) : Brick (7,3) at (5,12) ,3/14 totally

(AG-3) : Brick (7,11) at (7,13) ,4/14 totally

[ TW ]:I disappear brick at (7,12)
(AG-0) : Brick (11,3) at (9,5) ,2/14 totally

(AG-1) : Brick (11,5) at (10,5) ,3/14 totally

(AG-2) : Brick (11,7) at (12,11) ,2/14 totally

(AG-1) : Brick (11,14) at (13,2) ,4/14 totally

(AG-3) : Brick (14,8) at (13,9) ,5/14 totally
```

```
      [ TW ]:I appear brick at (0,0)
      (AG-2) : Brick (0,0) at (13,14) ,3/14 totally

      (AG-0) : Brick (14,10) destroyed

            ---(AG-3) time :12.00,  Score:0.357


            ---(AG-2) time :11.00,  Score:0.214


            ---(AG-0) time :11.00,  Score:0.143


            ---(AG-1) time :10.00,  Score:0.286

      [ TW ]:I appear obstacle at (0,0)
      [ TW ]:I appear hole at (0,1)
      [ TW ]:I appear hole at (0,2)
      [ TW ]:I appear brick at (0,3)
      [ TW ]:I disappear brick at (0,3)
      [ TW ]:I appear brick at (0,3)
      [ TW ]:I appear obstacle at (0,4)
      [ TW ]:I disappear obstacle at (0,0)
      [ TW ]:I disappear obstacle at (0,4)
      [ TW ]:I disappear brick at (0,3)
      [ TW ]:I appear hole at (0,0)
      [ TW ]:I appear brick at (0,3)
      [ TW ]:I disappear hole at (0,0)
```

Εκτέλεση 3.3β

```
      [ TW ]:I disappear hole at (1,3)
      [ TW ]:I disappear brick at (1,1)
      (AG-3) : Brick (3,3) at (2,3) ,1/15 totally

      (AG-1) : Brick (3,7) at (2,13) ,1/15 totally

      (AG-3) : Brick (4,1) at (3,1) ,2/15 totally

      (AG-1) : Brick (4,2) at (3,5) ,2/15 totally

      [ TW ]:I disappear hole at (3,6)
      (AG-2) : Brick (5,3) at (4,0) ,1/14 totally

(AG-0) : Brick (6,6) at (5,9) ,1/14 totally

(AG-3) : Brick (7,3) at (5,12) ,3/14 totally

(AG-3) : Brick (7,11) at (7,13) ,4/14 totally

[ TW ]:I disappear brick at (7,12)
(AG-0) : Brick (11,3) at (9,5) ,2/14 totally

(AG-1) : Brick (11,5) at (10,5) ,3/14 totally

(AG-2) : Brick (11,7) at (12,11) ,2/14 totally

(AG-1) : Brick (11,14) at (13,2) ,4/14 totally
```

```
(AG-3) : Brick (14,8) at (13,9) ,5/14 totally

[ TW ]:I appear brick at (0,0)
(AG-2) : Brick (0,0) at (13,14) ,3/14 totally

(AG-0) : Brick (14,10) destroyed

      ---(AG-1) time :11.00,  Score:0.286


      ---(AG-3) time :10.00,  Score:0.357


      ---(AG-0) time :10.00,  Score:0.143


      ---(AG-2) time :9.00,   Score:0.214

[ TW ]:I appear obstacle at (0,0)
[ TW ]:I appear hole at (0,1)
[ TW ]:I appear hole at (0,2)
[ TW ]:I appear brick at (0,3)
[ TW ]:I disappear brick at (0,3)
[ TW ]:I appear brick at (0,3)
[ TW ]:I appear obstacle at (0,4)
[ TW ]:I disappear obstacle at (0,0)
[ TW ]:I disappear obstacle at (0,4)
[ TW ]:I disappear brick at (0,3)
[ TW ]:I appear hole at (0,0)
[ TW ]:I appear brick at (0,3)
[ TW ]:I disappear hole at (0,0)

            * * *
```

Εκτέλεση 3.3γ

```
[ TW ]:I disappear hole at (1,3)
[ TW ]:I disappear brick at (1,1)
(AG-3) : Brick (3,3) at (2,3) ,1/15 totally

(AG-1) : Brick (3,7) at (2,13) ,1/15 totally

(AG-3) : Brick (4,1) at (3,1) ,2/15 totally

(AG-1) : Brick (4,2) at (3,5) ,2/15 totally

[ TW ]:I disappear hole at (3,6)
(AG-2) : Brick (5,3) at (4,0) ,1/14 totally

(AG-0) : Brick (6,6) at (5,9) ,1/14 totally

(AG-3) : Brick (7,3) at (5,12) ,3/14 totally

(AG-3) : Brick (7,11) at (7,13) ,4/14 totally

[ TW ]:I disappear brick at (7,12)
(AG-0) : Brick (11,3) at (9,5) ,2/14 totally

(AG-1) : Brick (11,5) at (10,5) ,3/14 totally
```

```
(AG-2) : Brick (11,7) at (12,11) ,2/14 totally

(AG-3) : Brick (11,14) at (13,2) ,5/14 totally

(AG-1) : Brick (14,8) at (13,9) ,4/14 totally

[ TW ]:I appear brick at (0,0)
(AG-2) : Brick (0,0) at (13,14) ,3/14 totally

(AG-0) : Brick (14,10) destroyed

      ---(AG-3) time :12.00,  Score:0.357


      ---(AG-1) time :10.00,  Score:0.286


      ---(AG-0) time :10.00,  Score:0.143


      ---(AG-2) time :9.00,   Score:0.214

[ TW ]:I appear obstacle at (0,0)
[ TW ]:I appear hole at (0,1)
[ TW ]:I appear hole at (0,2)
[ TW ]:I appear brick at (0,3)
[ TW ]:I disappear brick at (0,3)
[ TW ]:I appear brick at (0,3)
[ TW ]:I appear obstacle at (0,4)
[ TW ]:I disappear obstacle at (0,0)
[ TW ]:I disappear obstacle at (0,4)
[ TW ]:I disappear brick at (0,3)
[ TW ]:I appear hole at (0,0)
[ TW ]:I appear brick at (0,3)
[ TW ]:I disappear hole at (0,0)
```

* * *

Συμπεράσματα: Τετραπλασιάζοντας το φόρτο σε σχέση με την 3.1, παρατηρείται αύξηση λίγο παραπάνω από το τετραπλάσιο του χρόνου του μονοπρακτορικού συστήματος, Η αύξηση όμως των ενεργειών αυξάνει σημαντικά το χρόνο διεκπεραίωσης καθώς βλέπουμε στην εκτέλεση να παρεμβάλλονται πιο συχνά ενέργειες του κόσμου απ ο, τι πριν. Εξαιτίας αυτού ίσως οι πράκτορες έγιναν λιγότερο ανταγωνιστικοί όπως φαίνεται από τη διασπορά, την τυπική απόκλιση αλλά και το ποσοστό ενεργών πρακτόρων, καθώς μετά την εκτέλεση του κόσμου δίνεται ευκαιρία σε κάποιον να μπει στη συνάρτηση πράκτορα (ξεκλείδωμα σηματοφόρου).

Εκδοχή 4.1:

ΠΑΡΑΜΕΤΡΟΣ	ΤΙΜΗ
M	15
N	30
Τούβλα	9
Τρύπες	9
Εμπόδια	11
Ενέργειες TW (μέγιστο)	5
Πράκτορες	9

ΕΞΟΔΟΣ	ΤΙΜΗ
Μέσος χρόνος διεκπεραίωσης	11.814
Διασπορά χρόνου διεκπεραίωσης	22.541
Μέση βαθμολογία πράκτορα	0.111
Διασπορά βαθμολογίας	0.003
Τυπική απόκλιση βαθμολογίας	0.063
Ενεργοί πράκτορες (βαθμ. > 0)	77%
Ποσοστό (μη αδρανών) ενεργειών TW	80%

Εκτέλεση 4.1α

```
[ TW ]:I disappear hole at (1,3)
(AG-3) : Brick (3,3) at (2,3) ,1/8 totally

[ TW ]:I disappear hole at (3,5)
(AG-6) : Brick (4,1) at (4,0) ,1/7 totally

(AG-2) : Brick (4,2) at (5,9) ,1/7 totally

[ TW ]:I appear brick at (0,0)
[ TW ]:I disappear brick at (0,0)
(AG-7) : Brick (5,3) at (7,13) ,1/7 totally

(AG-0) : Brick (7,3) at (9,5) ,1/7 totally

(AG-4) : Brick (7,11) at (12,21) ,1/7 totally

(AG-1) : Brick (11,7) at (13,14) ,1/7 totally

(AG-2) : Brick (11,14) destroyed

      ---(AG-1) time :5.00,   Score:0.143


      ---(AG-2) time :5.00,   Score:0.143
```

```
(AG-8) : Brick (14,10) destroyed

      ---(AG-8) time :9.00,   Score:0.000

      ---(AG-6) time :9.00,   Score:0.143

      ---(AG-5) time :13.00,  Score:0.000

      ---(AG-3) time :13.00,  Score:0.143

      ---(AG-7) time :14.00,  Score:0.143

      ---(AG-0) time :16.00,  Score:0.143

      ---(AG-4) time :20.00,  Score:0.143
```

* * *

Εκτέλεση 4.1β

```
[ TW ]:I disappear hole at (1,3)
(AG-3) : Brick (3,3) at (2,3) ,1/8 totally

[ TW ]:I disappear hole at (3,5)
(AG-6) : Brick (4,1) at (4,0) ,1/7 totally

(AG-2) : Brick (4,2) at (5,9) ,1/7 totally

[ TW ]:I appear brick at (0,0)
[ TW ]:I disappear brick at (0,0)
(AG-7) : Brick (5,3) at (7,13) ,1/7 totally

(AG-0) : Brick (7,3) at (9,5) ,1/7 totally

(AG-4) : Brick (7,11) at (12,21) ,1/7 totally

(AG-1) : Brick (11,7) at (13,14) ,1/7 totally

(AG-2) : Brick (11,14) destroyed

      ---(AG-2) time :6.00,   Score:0.143

      ---(AG-1) time :5.00,   Score:0.143

(AG-8) : Brick (14,10) destroyed

      ---(AG-8) time :10.00,  Score:0.000

      ---(AG-6) time :9.00,   Score:0.143
```

```
      ---(AG-3) time :13.00,  Score:0.143

      ---(AG-5) time :13.00,  Score:0.000

      ---(AG-7) time :14.00,  Score:0.143

      ---(AG-0) time :16.00,  Score:0.143

      ---(AG-4) time :20.00,  Score:0.143
```

* * *

Εκτέλεση 4.1γ

```
[ TW ]:I disappear hole at (1,3)
(AG-3) : Brick (3,3) at (2,3) ,1/8 totally

[ TW ]:I disappear hole at (3,5)
(AG-6) : Brick (4,1) at (4,0) ,1/7 totally

(AG-2) : Brick (4,2) at (5,9) ,1/7 totally

[ TW ]:I appear brick at (0,0)
[ TW ]:I disappear brick at (0,0)
(AG-7) : Brick (5,3) at (7,13) ,1/7 totally

(AG-0) : Brick (7,3) at (9,5) ,1/7 totally

(AG-4) : Brick (7,11) at (12,21) ,1/7 totally

(AG-1) : Brick (11,7) at (13,14) ,1/7 totally

(AG-2) : Brick (11,14) destroyed

      ---(AG-2) time :6.00,   Score:0.143

      ---(AG-1) time :5.00,   Score:0.143

(AG-8) : Brick (14,10) destroyed

      ---(AG-6) time :10.00,  Score:0.143

      ---(AG-8) time :9.00,   Score:0.000

      ---(AG-5) time :14.00,  Score:0.000

      ---(AG-3) time :13.00,  Score:0.143

      ---(AG-7) time :15.00,  Score:0.143
```

```
---(AG-0) time :17.00,  Score:0.143

---(AG-4) time :20.00,  Score:0.143

                    * * *
```

Συμπεράσματα: Ο μέσος χρόνος διεκπεραίωσης είναι κατά πολύ μικρότερος από το εννιαπλάσιο του μονοπρακτορικού συστήματος, αλλά η διασπορά αυξήθηκε σημαντικά. Αυτό πιθανότατα οφείλεται στη μεγάλη ουρά που σχηματίζεται στο σηματοφόρο καθώς κάθε φορά 8 πράκτορες κλειδώνουν και μόνο ένας εκτελεί τις ενέργειες. Παρόλα αυτά φαίνεται πολύ μικρή διασπορά βαθμολογίας τυπική απόκλιση και αριθμός ενεργών πρακτόρων, καθώς υπάρχει κάποιος ανταγωνισμός μεταξύ τους αλλά όσοι είναι πάνω απ' το μηδέν πετυχαίνουν ίση βαθμολογία.

Εκδοχή 4.2:

ΠΑΡΑΜΕΤΡΟΣ	ΤΙΜΗ
M	15
N	15
Τούβλα	3
Τρύπες	3
Εμπόδια	11
Ενέργειες TW (μέγιστο)	8
Πράκτορες	9

ΕΞΟΔΟΣ	ΤΙΜΗ
Μέσος χρόνος διεκπεραίωσης	11.666
Διασπορά χρόνου διεκπεραίωσης	29.058
Μέση βαθμολογία πράκτορα	0.111
Διασπορά βαθμολογίας	0.104
Τυπική απόκλιση βαθμολογίας	0.323
Ενεργοί πράκτορες (βαθμ. > 0)	11%
Ποσοστό (μη αδρανών) ενεργειών TW	63%

Εκτέλεση 4.2α

```
      [ TW ]:I disappear hole at (2,2)
(AG-3) : Brick (5,3) at (13,6) ,1/2 totally
```

```
[ TW ]:I disappear hole at (14,0)
(AG-6) : Brick (11,14) destroyed
(AG-2) : Brick (14,10) destroyed

      ---(AG-2) time :5.00,   Score:0.000


      ---(AG-6) time :5.00,   Score:0.000


      ---(AG-7) time :6.00,   Score:0.000


      ---(AG-0) time :8.00,   Score:0.000


      ---(AG-4) time :13.00,  Score:0.000


      ---(AG-1) time :12.00,  Score:0.000

[ TW ]:I appear brick at (0,0)
[ TW ]:I disappear brick at (0,0)

      ---(AG-8) time :14.00,  Score:0.000


      ---(AG-5) time :19.00,  Score:0.000


      ---(AG-3) time :18.00,  Score:1.000

[ TW ]:I appear obstacle at (0,0)

            * * *
```

Εκτέλεση 4.2β

```
[ TW ]:I disappear hole at (2,2)
(AG-3) : Brick (5,3) at (13,6) ,1/2 totally

[ TW ]:I disappear hole at (14,0)
(AG-6) : Brick (11,14) destroyed
(AG-2) : Brick (14,10) destroyed

      ---(AG-2) time :6.00,   Score:0.000


      ---(AG-6) time :5.00,   Score:0.000


      ---(AG-7) time :7.00,   Score:0.000


      ---(AG-0) time :9.00,   Score:0.000


      ---(AG-1) time :14.00,  Score:0.000
```

```
      ---(AG-4) time :14.00,  Score:0.000

[ TW ]:I appear brick at (0,0)
[ TW ]:I disappear brick at (0,0)

      ---(AG-8) time :16.00,  Score:0.000


      ---(AG-3) time :20.00,  Score:1.000


      ---(AG-5) time :19.00,  Score:0.000

[ TW ]:I appear obstacle at (0,0)

            * * *
```

Συμπεράσματα: Σε σχέση με πριν αυξήθηκε η διασποράς του χρόνου διεκπεραίωσης, κάτι που ενδεχομένως οφείλεται στην εκ των υστέρων άσκοπη αναζήτηση των πρακτόρων που προλαβαίνουν να πάρουν τούβλα για κάποια τρύπα εξαιτίας του πολύ μικρού φόρτου. Ομοίως η διασπορά και η τυπική απόκλιση της βαθμολογίας αυξήθηκαν σημαντικά καθώς μόνο ένας πράκτορας από τους εννιά καταφέρνει να γεμίσει τη μία τρύπα που μένει, άρα το σύστημα γίνεται πολύ πιο ανταγωνιστικό. Επιπλέον ο κόσμος παρεμβάλλεται περισσότερο στις εκτελέσεις των πρακτόρων

Εκδοχή 4.3:

ΠΑΡΑΜΕΤΡΟΣ	ΤΙΜΗ
M	15
N	15
Τούβλα	12
Τρύπες	9
Εμπόδια	6
Ενέργειες TW (μέγιστο)	10
Πράκτορες	9

ΕΞΟΔΟΣ	**ΤΙΜΗ**
Μέσος χρόνος διεκπεραίωσης	12.815
Διασπορά χρόνου διεκπεραίωσης	27.156
Μέση βαθμολογία πράκτορα	0.101
Διασπορά βαθμολογίας	0.003
Τυπική απόκλιση βαθμολογίας	0.055
Ενεργοί πράκτορες (βαθμ. > 0)	78%
Ποσοστό (μη αδρανών) ενεργειών TW	40%

Εκτέλεση 4.3α

```
[ TW ]:I disappear hole at (1,3)
(AG-3) : Brick (1,1) at (2,3) ,1/8 totally

[ TW ]:I disappear hole at (3,5)
(AG-6) : Brick (3,3) at (4,0) ,1/7 totally

(AG-2) : Brick (4,1) at (5,9) ,1/7 totally

[ TW ]:I appear brick at (0,0)
[ TW ]:I disappear brick at (0,0)
(AG-7) : Brick (4,2) at (7,13) ,1/7 totally

(AG-0) : Brick (5,3) at (9,5) ,1/7 totally

(AG-4) : Brick (6,6) at (13,6) ,1/7 totally

(AG-1) : Brick (7,3) at (13,14) ,1/7 totally

(AG-2) : Brick (7,11) destroyed

      ---(AG-1) time :5.00,   Score:0.143


      ---(AG-2) time :5.00,   Score:0.143

[ TW ]:I appear obstacle at (0,0)
(AG-8) : Brick (11,7) destroyed
(AG-6) : Brick (11,14) destroyed

      ---(AG-8) time :9.00,   Score:0.000


      ---(AG-6) time :9.00,   Score:0.143

(AG-3) : Brick (14,10) destroyed

      ---(AG-3) time :13.00,  Score:0.143


      ---(AG-5) time :13.00,  Score:0.000

[ TW ]:I appear brick at (0,2)
```

```
(AG-7) : Brick (0,2) destroyed

      ---(AG-7) time :14.00,  Score:0.143


      ---(AG-0) time :17.00,  Score:0.143

[ TW ]:I appear obstacle at (0,3)

      ---(AG-4) time :18.00,  Score:0.143

[ TW ]:I appear hole at (0,4)
[ TW ]:I appear hole at (0,5)

                  * * *
```

Εκτέλεση 4.3β

```
[ TW ]:I disappear hole at (1,3)
(AG-3) : Brick (1,1) at (2,3) ,1/8 totally

[ TW ]:I disappear hole at (3,5)
(AG-6) : Brick (3,3) at (4,0) ,1/7 totally

(AG-2) : Brick (4,1) at (5,9) ,1/7 totally

[ TW ]:I appear brick at (0,0)
[ TW ]:I disappear brick at (0,0)
(AG-7) : Brick (4,2) at (7,13) ,1/7 totally

(AG-0) : Brick (5,3) at (9,5) ,1/7 totally

(AG-4) : Brick (6,6) at (13,6) ,1/7 totally

(AG-1) : Brick (7,3) at (13,14) ,1/7 totally

(AG-2) : Brick (7,11) destroyed

      ---(AG-2) time :6.00,   Score:0.143


      ---(AG-1) time :6.00,   Score:0.143

[ TW ]:I appear obstacle at (0,0)
(AG-8) : Brick (11,7) destroyed
(AG-6) : Brick (11,14) destroyed

      ---(AG-8) time :10.00,  Score:0.000


      ---(AG-6) time :10.00,  Score:0.143

(AG-3) : Brick (14,10) destroyed

      ---(AG-3) time :15.00,  Score:0.143


      ---(AG-5) time :14.00,  Score:0.000

[ TW ]:I appear brick at (0,2)
```

```
(AG-7) : Brick (0,2) destroyed

      ---(AG-7) time :16.00,  Score:0.143


      ---(AG-0) time :19.00,  Score:0.143

[ TW ]:I appear obstacle at (0,3)

      ---(AG-4) time :22.00,  Score:0.143

[ TW ]:I appear hole at (0,4)
[ TW ]:I appear hole at (0,5)

            * * *
[ TW ]:I disappear hole at (1,3)
(AG-3) : Brick (1,1) at (2,3) ,1/8 totally

[ TW ]:I disappear hole at (3,5)
(AG-6) : Brick (3,3) at (4,0) ,1/7 totally

(AG-2) : Brick (4,1) at (5,9) ,1/7 totally

[ TW ]:I appear brick at (0,0)
[ TW ]:I disappear brick at (0,0)
(AG-7) : Brick (4,2) at (7,13) ,1/7 totally

(AG-0) : Brick (5,3) at (9,5) ,1/7 totally

(AG-4) : Brick (6,6) at (13,6) ,1/7 totally

(AG-1) : Brick (7,3) at (13,14) ,1/7 totally

(AG-2) : Brick (7,11) destroyed

      ---(AG-1) time :6.00,   Score:0.143


      ---(AG-2) time :5.00,   Score:0.143

[ TW ]:I appear obstacle at (0,0)
(AG-8) : Brick (11,7) destroyed
(AG-6) : Brick (11,14) destroyed

      ---(AG-8) time :11.00,  Score:0.000


      ---(AG-6) time :11.00,  Score:0.143

(AG-3) : Brick (14,10) destroyed

      ---(AG-3) time :14.00,  Score:0.143


      ---(AG-5) time :14.00,  Score:0.000

[ TW ]:I appear brick at (0,2)
(AG-7) : Brick (0,2) destroyed

      ---(AG-7) time :16.00,  Score:0.143
```

```
      ---(AG-0) time :18.00,  Score:0.143

[ TW ]:I appear obstacle at (0,3)

      ---(AG-4) time :20.00,  Score:0.143

[ TW ]:I appear hole at (0,4)
[ TW ]:I appear hole at (0,5)

[ TW ]:I disappear hole at (1,3)
```

* * *

Εκτέλεση 4.3γ

```
(AG-3) : Brick (1,1) at (2,3) ,1/8 totally

[ TW ]:I disappear hole at (3,5)
(AG-6) : Brick (3,3) at (4,0) ,1/7 totally

(AG-2) : Brick (4,1) at (5,9) ,1/7 totally

[ TW ]:I appear brick at (0,0)
[ TW ]:I disappear brick at (0,0)
(AG-7) : Brick (4,2) at (7,13) ,1/7 totally

(AG-0) : Brick (5,3) at (9,5) ,1/7 totally

(AG-4) : Brick (6,6) at (13,6) ,1/7 totally

(AG-1) : Brick (7,3) at (13,14) ,1/7 totally

(AG-2) : Brick (7,11) destroyed

      ---(AG-1) time :6.00,   Score:0.143


      ---(AG-2) time :5.00,   Score:0.143

[ TW ]:I appear obstacle at (0,0)
(AG-8) : Brick (11,7) destroyed
(AG-6) : Brick (11,14) destroyed

      ---(AG-8) time :11.00,  Score:0.000


      ---(AG-6) time :11.00,  Score:0.143

(AG-3) : Brick (14,10) destroyed

      ---(AG-3) time :14.00,  Score:0.143


      ---(AG-5) time :14.00,  Score:0.000

[ TW ]:I appear brick at (0,2)
(AG-7) : Brick (0,2) destroyed
```

```
      ---(AG-7) time :16.00,  Score:0.143


      ---(AG-0) time :18.00,  Score:0.143

[ TW ]:I appear obstacle at (0,3)

      ---(AG-4) time :20.00,  Score:0.143

[ TW ]:I appear hole at (0,4)
[ TW ]:I appear hole at (0,5)

            * * *
```

Συμπεράσματα: Παρατηρείται σημαντική μείωση των μη-αδρανών ενεργειών που ενδεχομένως να σχετίζεται με τη μεγαλύτερη εμπλοκή του κόσμου ανάμεσα στην εκτέλεση των πρακτόρων. Επίσης το κάπως μεγαλύτερο φόρτο επιτρέπει στους πράκτορες να δουλεύουν πιο συνεργατικά από πριν.

Γενικά συμπεράσματα

- Όσο η διασπορά και η τυπική απόκλιση βαθμολογίας μειώνεται τόσο λιγότερο ανταγωνιστικοί είναι οι πράκτορες
- Η μείωση διασποράς μέσου χρόνου διεκπεραίωσης ερμηνεύεται ως εξίσου ατυχείς επιλογές για τους πράκτορες (μακρινά τούβλα, διαδρομές με πολλά εμπόδια κλπ)
- Η ανταγωνιστικότητα είναι ανάλογη του φόρτου και αντιστρόφως ανάλογη του αριθμού πρακτόρων.
- Το κέρδος σε σχέση με τις εκδοχές λιγότερων πρακτόρων εξαρτάται απ τη μη-ανταγωνιστικότητα των πρακτόρων.
- Αύξηση των μη αδρανών ενεργειών του κόσμου συνεπάγεται αύξηση του μέσου χρόνου διεκπεραίωσης
- Περισσότερες αδρανείς ενέργειες παρατηρούνται σε εκδοχές με μεγάλη εμπλοκή του κόσμου στις εκτελέσεις των πρακτόρων.
- Ο αριθμός των πρακτόρων στο σύστημα είναι ανάλογος του κέρδους χρόνου
- Η πυκνότητα του κόσμου σε τούβλα και τρύπες είναι σχεδόν ανάλογη του πόσο μεγάλες αποστάσεις γίνονται
- Ο πράκτορας δεν αλληλεπιδρά πολύ με τον κόσμο («προτίμηση» νεοεμφανισθέντων στοιχείων)
- Το μέγεθος του κόσμου δεν επηρεάζει με συγκεκριμένο τρόπο τις εκτελέσεις

Βιβλιογραφία - Παραπομπές

1) Θεωρία πρακτόρων
Εισαγωγή στα πολυπρακτορικά συστήματα
(Michael Wooldridge , μετάφραση Δασκαλοπούλου Ασπασία , εκδόσεις Κλειδάριθμος)

Δασκαλοπούλου Ασπασία
Διαφάνειες μαθήματος «Συστήματα Πρακτόρων»
http://inf-server.inf.uth.gr/courses/CE454/

2) Θεωρητικά ζητήματα ταυτόχρονου προγραμματισμού
Λάλης Σπυρίδων
Διαφάνειες μαθήματος Προγραμματισμός ΙΙΙ
http://inf-server.inf.uth.gr/courses/CE220/

3) Υπολογισμός στατιστικών για κάθε εκδοχή
http://easycalculation.com/statistics/standard-deviation.php

4) Θεωρητικά στοιχεία για προβλήματα λαβυρίνθου
http://en.wikipedia.org/wiki/Maze_solving_algorithm

5) Αλγόριθμοι αναζήτησης
Τεχνητή Νοημοσύνη Β' Έκδοση
(Βλαχάβας Ιωάννης, Κεφάλας Πέτρος, Βασιλειάδης Νικόλαος, Κόκκορας Φώτης, Σακελλαρίου Ηλίας, Εκδόσεις Γαρταγάνη)

6) Αρχεία υλοποίησης συναρτήσεων ουράς (queue.c , queue.h)
Simon Howard
http://c-algorithms.sourceforge.net/