Annals of DAAAM for 2012 & Proceedings of the 23rd International DAAAM Symposium, Volume 23, No.1, ISSN 2304-1382
ISBN 978-3-901509-91-9, CDROM version, Ed. B. Katalinic, Published by DAAAM International, Vienna, Austria, EU, 2012
Make Harmony between Technology and Nature, and Your Mind will Fly Free as a Bird
Annals & Proceedings of DAAAM International 2012

SIMULTANEOUS COORDINATION OF TWO ROBOTS WITH OVERLAPPING WORKSPACES

CURKOVIC, P[etar]; DUBIC, K[runoslav]; JERBIC, B[ojan] & STIPANCIC, T[omislav]

Abstract: *The paper presents a novel approach to path planning for two decoupled robots with six degrees of freedom (DOF) each. The robots are placed so that their workspaces overlap. Thus each robot presents dynamic obstacle to the other one, whereas each robot is controlled with separate controller in a separated time domain. The method is based on analysis of the configuration spaces (C space) of the robots. For given initial and final configurations of the robots, considering a set of constraints, the algorithm successfully finds a set of intermediate configurations to ensure smooth, collision free transition from start to end configuration. The algorithm is tested in Fanuc's physical simulation environment Roboguide, and successfully experimentally implemented on two Fanuc's LR Mate 200iC/5L robots. The scenario is applied for assembly task of industrial thermo regulator, with assembly parts randomly distributed within the shared spaces of the two robots*

1. INTRODUCTION

To increase manipulability of a technical system, increased number of the *DOF* is required. For example, human hand, has 27 *DOF* in total, with 19 *DOF* excluding the wrist. The complexity of building appropriate, or similar, technical system increases as well as necessary computational power. Moreover, humans and all higher primates have symmetrically distributed limbs. This is a result of a long term evolutionary process.

In this paper, an *anthropomatic* system is built of well known, two six *DOF* robots that work with a certain level of coordination in circumstances where the workspaces of the robots overlap. The reason for such a setup is to increase the performance of the system compared to capabilities two independent six *DOF* robots each.

Standard industrial approach to programming robotic systems where the workspaces overlap is to ensure that the robots actually never enter the area of interference at the same time.

This can be achieved by tracking the position of a *master* robot, and force the *slave* robot to adapt its movement not to interfere with the *master*. It becomes immediately evident that such an approach results with time delays. Moreover it is an unnatural way of motion execution, with lot of pauses, stop and go intermittent motions etc.

We propose a simple but intuitive scenario. The two robots are arranged as shown in Fig. 1. There are a number of objects of interest present in the workspaces of the robots. The robots have to recognize the objects, pick them up and deliver them to a desired location. The robots are working simultaneously, thus presenting dynamic obstacle to each other.

A comprehensive survey on robot motion planning can be found in [1].

An approach to a path planning of a dual-arm reconfigurable robot is presented in [2]. The problem deals with a SCARA-like configuration of a robot controlled by a single controller. Real-time optimum is achieved for given start and end positions of the tool centre point (*TCP*). The algorithms were successfully applied to an industrial application of gear assembly, with some limitations in terms of reliability.

There are a number of research approaches to compliance control of a coupled dual arm robotic systems. Such systems are based on two robots which have a contact between reference points, either by caring a common object or directly. These methods are based on impedance control, to maintain interaction forces within some predefined level [3].

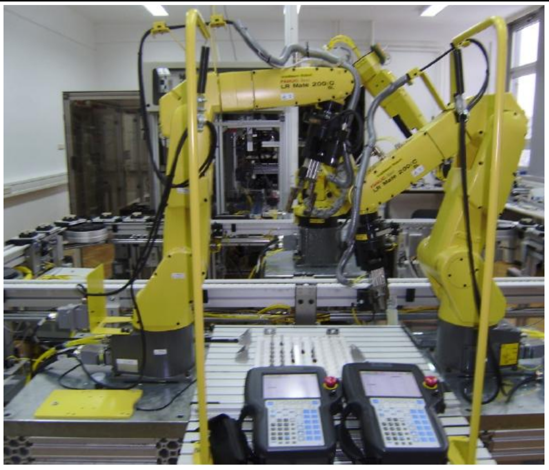


Fig. 1. Two robots with overlapping workspaces used as a model for collision free planning.

Another approach is to employ artificial intelligence based methods to find solution of the problem of path planning. There are approaches based on expert systems, artificial neural networks. There are issues with completeness of solutions given by the algorithms as well as the time required to find an optimal or near-optimal solution [4].

If the problem of path planning is represented in optimization context, robust optimization techniques, such as evolutionary algorithms, swarm intelligence concepts [5-10] etc. have proven suitable since the problem is NP-complete even in iys simplest form.

In this paper we proposed a method based on innovative approach of C-space reduction, that enables on-line path planning, while robots are moving in *point to point* (*PTP*) mode. The inputs to the robots are initial and final locations of the *TCP*. The algorithm checks the feasibility of input data, calculates the necessary intermediate going – through configurations and delivers the data to the robot controller.

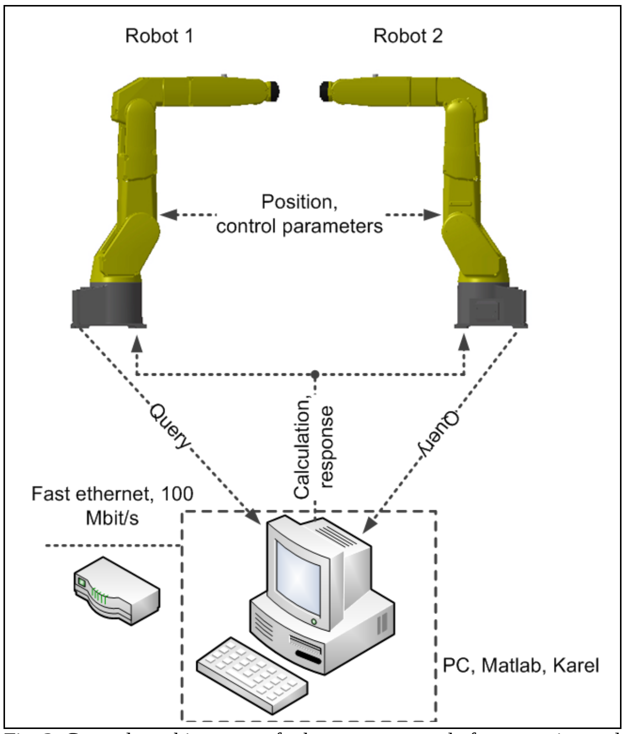


Fig. 2. General architecture of the system used for experimental evaluation

2. FEATURES OF THE SYSTEM

The system used in this paper is composed of following components: two Fanuc Lr Mate 500 iC robots, equipped with fast Ethernet cards, force / torque sensors, cameras, Karel programming language, as presented with Fig. 2.

The robots are connected via fast Ethernet to each other and to a PC, which has Matlab R2008 installed. The Matlab is extended with the MuPAD symbolic engine, optimized for operating on symbolic mathematical expressions and customized plotting.

2.1 Planning algorithm

The calculation begins after the robots receive information on the start and end configurations. These data is provided from the robot controller as a query message to the PC.

First a case where the initial and final configurations are provided by the human operator is considered. At a later stage these data is provided as a variable from the vision system attached to the robots.

Matlab checks whether initial and final configurations result with collision of arbitrary part of the robot 1 and robot 2. If there is collision present, the planning cannot be executed and the error message is provided indicating that different set of initial conditions has to be provided.

If initial conditions are acceptable, the planning procedure is executed in Matlab. The output of this procedure are going – through points, reference points that are provided to the robot for motion execution, see Fig. 3.

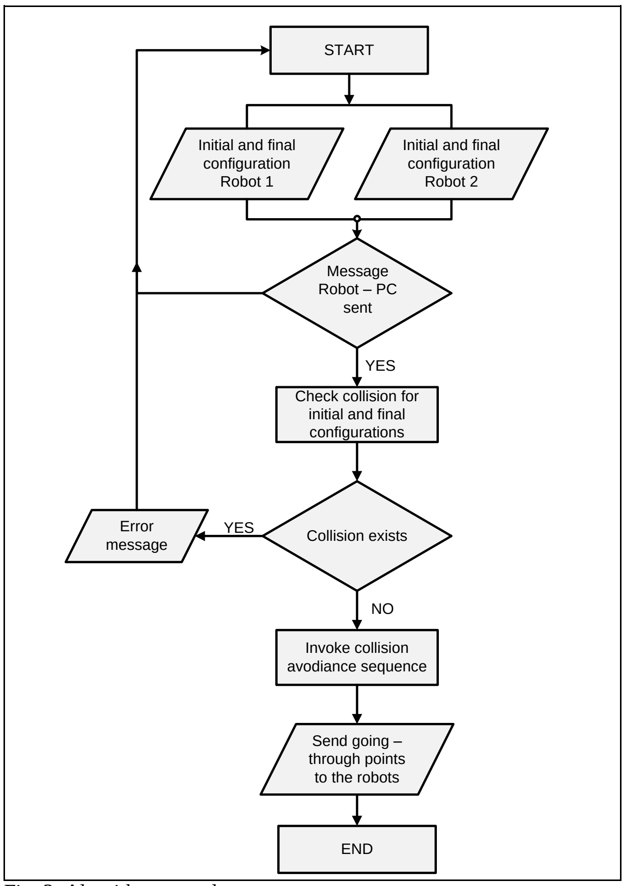

START
Initial and final configuration Robot 1
Initial and final configuration Robot 2
Message Robot – PC sent
YES
Check collision for initial and final configurations
Error message
YES
Collision exists
NO
Invoke collision avodiance sequence
Send going – through points to the robots
END

Fig. 3. Algorithm procedure

2.2 Static case

Motion is executed in point to point (*PTP*) mode for exactness of the trajectory and safety. *PTP* motion in reality means that the robot has to stop for some finite amount of time to change the path-defining vector. This time in our experiment is <0.1 s.

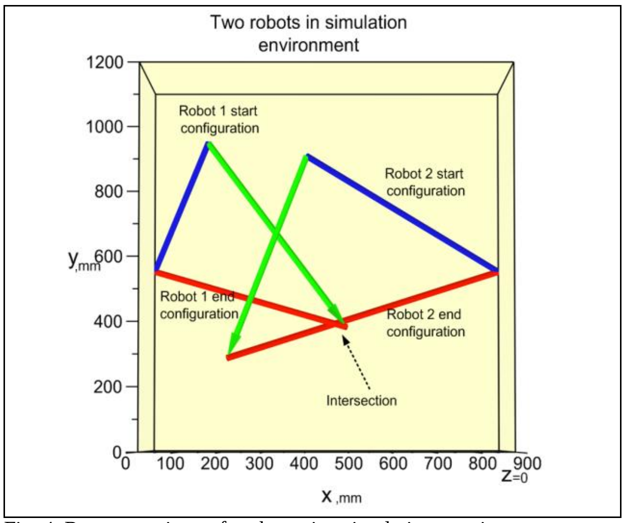


Fig. 4. Representation of robots in simulation environment – no solution

It is possible to interpolate the points with a spatial curve to get the smooth trajectory and fluid motion of the robot. The problem is how to decide to which accuracy the interpolation should be allowed, to avoid any possibility of collision.

This is an important problem whose analysis is out of the scope of this paper.

Robots are represented in Matlab as a set of lines whose intersections are analysed. If an intersection is present, the trajectory should be modified to enable collision free movement. Since the model is simplified with a set of lines, whereas real robots have defined physical dimensions, a parameter is experimentally determined to artificially increase the distance between the segments of the robots. By changing size of this parameter, it is possible to influence the actual proximity of the critical robot segments passing each other. This also enables compensation of different tools that can be attached to the flange of the robots thus changing the geometry.

The simulation environment corresponds to the top view of the robots and analysis taking place in the horizontal plane, which is defined with the bases of the two robots.

This means that the planning is reduced to 2*D* problem, considering the *TCP* as reference point, defined in *XY* plane.

While planning the trajectory, each robot considers the other robot a static obstacle that should be avoided. For finding a collision free path, a method based on *visibility graph* [4] approach is employed. In this method, a non-directed graph *G* is constructed in the *C*-space with polygonal obstacles. The nodes of the graph are the initial and goal configurations and all the obstacle vertices. The links of *G* are all the straight line segments connecting two nodes that do not intersect the interior of the obstacle region. The algorithm developed in this paper finds the shortest subgraph of *G*, whose nodes present going through points of the robots *TCP*.

Fig. 4 presents a setup of robots where no solution exists due to the end configurations of the two robots. The end configurations result in collision between the segments of the robots and the error message with the appropriate warning is delivered prior to motion planning.

Fig 5. illustrates a similar problem, with existing solution and the solution is provided by the algorithm in form of one *going through* points that are delivered as a list of variables to the controller of the robot 1. In this particular case, robot 1 (left robot) is avoiding the robot 2, because the motion of the robot 1 envelops the complete motion of the robot 2. Robot 2 could avoid the robot 1 by waiting for some amount of time, or by modulation of its *TCP* speed. This leads to a time – dependant problem domain.

The algorithm was at this point organized to perform the planning on the beginning of the motion, and no further planning was possible. This can be considered as a limitation, because no replanning due to some changes in environment can be executed. For example, if one robot unexpectedly stops, the result will be a collision. Although it is unlikely that a robot stops for no reason, there is another reason why dynamic, time dependant planning, is a very a useful feature

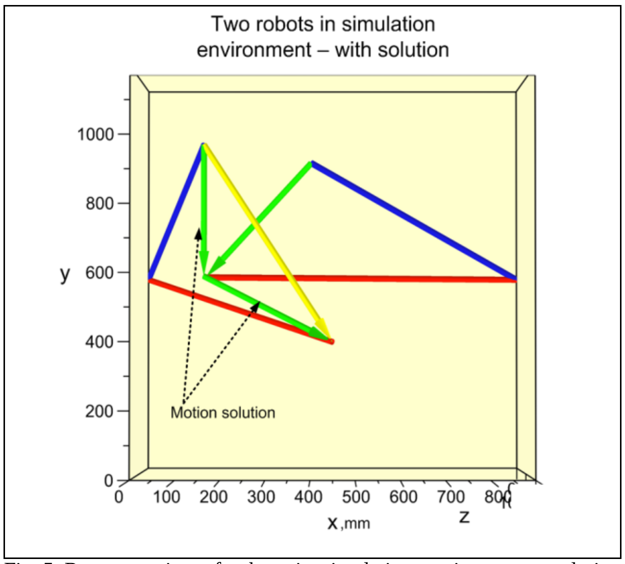


Fig. 5. Representation of robots in simulation environment– solution provided in form of trajectory

2.2 Dynamic case

Planning in advance limits the optimality of given trajectory in terms of the trajectory length, an that has a direct effect to the time required to fulfil a task. The reason for that is the fact that one robot might already leave the area in workspace where potential collision might occur – as in the scenario presented on Fig. 6, while the other robot, not knowing the position of the first robot in each time step, is making unnecessary loop in the workspace trying to avoid the possibility for collision.

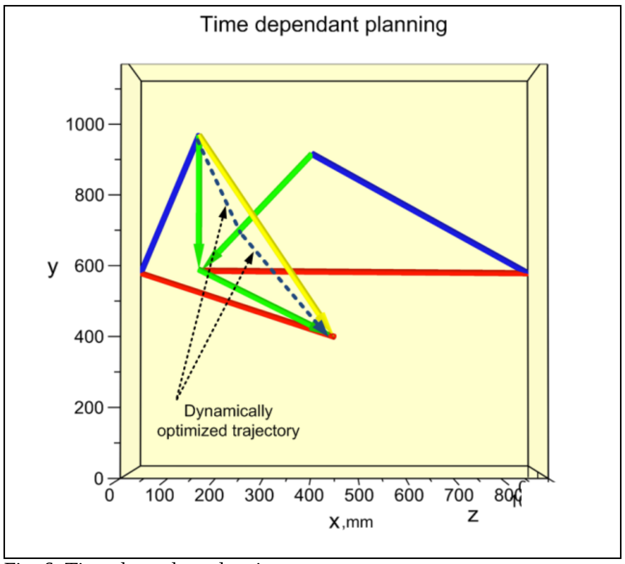
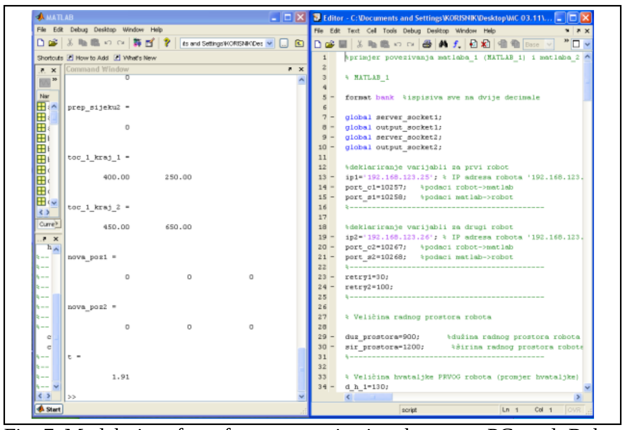


Fig. 6. Time dependent planning

It is not possible to notice this effect in the simulation environment since the trajectories are delivered at once with no time dependent variables. It was only after the implementation on the real robots that we have realized that robots are making unnecessarily large deviations from the optimal trajectories.

That is the reason why a method is developed that enables the time – dependent planning, with motion compensation. It is important to notice that all procedures are executed on the fly, in real time on real industrial equipment installed in our laboratory, 100 Mbit/s fast Ethernet, PC with 1.86 GHz Intel Core 2 Duo and 2 GB RAM, the planning together with plotting on the screen required less then 0.1s.

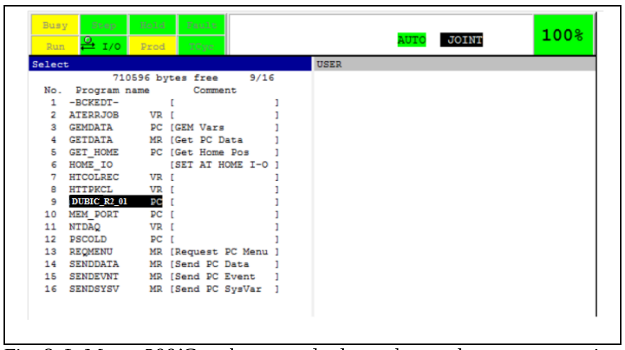
This time has proven acceptable for the real application, tested on our laboratory setup.


MATLAB
File Edit Debug Desktop Window Help
Shortcuts How to Add What's New
Command Window
Start
Editor - C:\Documents and Settings\KORISNIK\Desktop\MC 03.11\...
File Edit Text Cell Tools Debug Desktop Window Help
script
Ln 1 Col 1

```
       0

prep_sijeku2 =

       0

toc_1_kraj_1 =

     400.00      250.00

toc_1_kraj_2 =

     450.00      650.00

nova_poz1 =

       0           0           0

nova_poz2 =

       0           0           0

t =

     1.91

>>
```

```
%primjer povezivanja matlaba_1 (MATLAB_1) i matlaba_2

% MATLAB_1

format bank  %ispisiva sve na dvije decimale

global server_socket1;
global output_socket1;
global server_socket2;
global output_socket2;

%deklariranje varijabli za prvi robot
ip1='192.168.123.25'; % IP adresa robota '192.168.123.
port_c1=10257;   %podaci robot->matlab
port_s1=10258;   %podaci matlab->robot
%-----------------------------------------

%deklariranje varijabli za drugi robot
ip2='192.168.123.26'; % IP adresa robota '192.168.123.
port_c2=10267;   %podaci robot->matlab
port_s2=10268;   %podaci matlab->robot
%-----------------------------------------
retry1=30;
retry2=100;
%-----------------------------------------

% Veličina radnog prostora robota

duz_prostora=900;      %dužina radnog prostora robota
sir_prostora=1200;      %širina radnog prostora robote
%-----------------------------------------

% Veličina hvataljke PRVOG robota (promjer hvataljke)
d_h_1=130;
```

Fig. 7. Matlab interface for communication between PC and Robot controller



```
          710596 bytes free       9/16
 No.  Program name       Comment
  1  -BCKEDT-          [                 ]
  2  ATERRJOB      VR  [                 ]
  3  GEMDATA       PC  [GEM Vars         ]
  4  GETDATA       MR  [Get PC Data      ]
  5  GET_HOME      PC  [Get Home Pos     ]
  6  HOME_IO           [SET AT HOME I-O  ]
  7  HTCOLREC      VR  [                 ]
  8  HTTPKCL       VR  [                 ]
  9  DUBIC_R2_01   PC  [                 ]
 10  MEM_PORT      PC  [                 ]
 11  NTDAQ         VR  [                 ]
 12  PSCOLD        PC  [                 ]
 13  REQMENU       MR  [Request PC Menu  ]
 14  SENDDATA      MR  [Send PC Data     ]
 15  SENDEVNT      MR  [Send PC Event    ]
 16  SENDSYSV      MR  [Send PC SysVar   ]
```

Fig. 8. LrMate 200*i*C robot standard teach pendant programming interface

For dynamic planning several intermediate checks are performed, after initial planning. The scenario is similar to the static, forward – planning case, but planning procedure is evoked more than once, in a number of discrete time steps.

After the algorithm determines which robots envelope covers the others robot envelope, the *TCP* of the enveloped robot becomes a dynamic node in the *G* graph. The position of this node is monitored in *t* discrete time steps. This enables the robots to start the movement simultaneously, and to modulate the trajectory for better optimization in terms of total path traveled and time elapsed, as illustrated with Fig. 6.

The number of discrete time steps *t* for the dynamic calculation influences the exactness and optimality of the calculated trajectory. The trade off is given in time required for the intermediate recalculations. While calculating, the robot stops for a finite time, approximately 0.1 *s*, what becomes notable in robot movement as a reference going through point in point to point movement. This effect is emphasized if the speed of the *TCP* increases.

3. CONCLUSIONS

A method based on visibility graph and *C* space reduction proved suitable for real time coordination of two industrial 6 *DOF* robots with overlapping workspaces.

The proposed algorithms are successfully implemented on an assembly setup, as a low-level real time background procedure. This ensures faster, optimized robot motion, eliminating at the same time possibility of collision between moving segments of the robots. At this point the model is simplified and the planning is conducted in one plane. The motivation for future work is to expand the model to full space to enable full usability and flexibility of the dualarm robotic system. Fig 7. and Fig. 8 present current state of the communication interface that has to be activated prior to planning execution. At this point it is unintuitive and not user friendly, since the operator has to define a set of parameters by hand running both Matlab and the programming interface of the robot. In future version of the programme a new interface will be implemented to ease the operation of the whole system.

4. ACKNOWLEDGEMENTS

The authors kindly acknowledge the support of the Croatian Ministry of sciences, education and sport through project No. 120-1201948-1941*Autonomous multi-agent automated assembly* and technological project TP-46 *Robotics applications in neurosurgery*. Note that this paper is written in coordination with student KrunoslavDubic.

5. REFERENCES

[1] Latombe, J-C. (1991).*Robot Motion Planning*. Kluwer Academic Publishers. Boston

[2] Fey, Y., Fuqiuang, D., Xifang, Z., (2004) Collision-free motion planning of dual-arm reconfigurable robots, *Robotics and Computer-Integrated Manufacturing* 20, p351-357

[3] Surdilovic, D., Yakut, Y., Nguyen, T-M., Pham, X. B., Vick, A., Martin, M., (2010). Compliance Control with Dual-Arm Humanoid Robots: Design, Planning and Programming. “010 IEEE-RAS *International conference on Humanoid Robots, Nashville*, TN, USA

[4] Ćurković, P.; Jerbić, B.; Stipančić, T. (2008) Hybridization of adaptive genetic algorithm and ART 1 neural architecture for efficient path planning of a mobile robot. Trans of FAMENA 32, pp 11-21

[5] Za'er S. Abo-H.; Othman A.; Sofian I. B, Al-Omari, M. Nafee' A. (2011). Continuous Genetic Algorithms for Collision-Free Cartesian Path Planning of Robot Manipulators, Int. Journ. of Adv. Rob. Sys., Vol. 6

[6] Ćurković, P., Jerbić, B. (2007). Honey-bees optimization algorithm applied to path planning problem. *International journal of simulation modeling.* VI, pp.154-165

[7] Curkovic, P., Jerbic, B. & Stipancic, T. (2012). Coordination of robots with overlapping workspaces based on motion co-evolution. *International journal of simulation modeling.* In press

[8] Venegas Montes, H.A., Raymundo Marcial-Romero, J. (2009). An Evolutionary Path Planner for Multiple Robot Arms. In: *Evo Workshops*, LNCS, Springer, Heidelberg

[9] Solteiro Pires, E.J., Tenreiro Machado, J.A., Moura Oliveira, P.B. (2004). Robot Trajectory Planning Using Multi-objective Genetic Algorithm Optimization. Proc- of the GECCO, LNCS

[10] Nearchou, A. (1998) Path planning robot using genetic heuristics. Robotica, vol.16, pp.575-588