Application of Simulated Annealing to Dexterous Object Manipulation with Multiple Agents

Gábor Vass * Shahram Payandeh † Béla Lantos *

* Technical University of Budapest
Department of Control Engineering and Information Technology
H-1117 Budapest, Pázmány Péter sétány 1/D. Hungary

† Simon Fraser University, Experimental Robotics Laboratory, School of Engineering Science
8888 University Drive, Burnaby, British Columbia Canada V5A 1S6

Abstract

The manipulation task (called object re-configuration problem) is stated as the following: given an initial grasp of the object find the motion's trajectories of the agents to move the object to the desired configuration. In general collision free paths for all agents must be found toward the contact points on the object (pre-grasp configuration) and the grasping and manipulation forces should then be exerted on the object by the agents. These forces are determined first to ensure a stable grasp, then to manipulate the object. We present a model based motion planner algorithm for manipulating agents for object re-configuration using simulated annealing (SA) algorithm for generating the relative motion between the object and agents. The motion of the agents relative to the object can be pure sliding, pure rolling, breaking contact and agent relocation is allowed. The motion sequence is represented by a relative velocity matrix. The algorithm can be used for example to move a known shaped object to a different position and orientation with a robotic hand. Simulation results are presented.

1 Introduction

Object manipulation either by multiple agent or dexterous robotic hand is a center field in robotic research. In our paper a motion planner for object manipulation with multiple dextrous agents is proposed.

A survey of robot grasp synthesis algorithms is described in [1]. The object manipulation problem can be divided into two sub-problems: the global planner and the local planner [2, 3]. The task of the global planner is to search for nominal path in the configuration space among static obstacles between the initial and the desired position and orientation of the object by generating points (subgoals) in the object's configuration space, the local planner tries to find admissible motion of the agents between the subgoals. The detailed trajectories of the agents' motion (path) between two subgoals are generated by the local planner.

Numerous object manipulation algorithms have been proposed, one of the most frequent model used in motion planner algorithms is the description of the motion of two contacting object which is given in [5]. An algorithm for four-finger equilibrium and force-closure grasps of polyhedral objects is proposed by [6], another algorithm for dextrous grasping force optimization is given by [7].

The topic of object manipulation with agents or robotic hands can be classified by the mode of the relative motion of the agents and the object. Pushing, rolling and sliding are examples of relative motions which the agents can produce with respect to the object. In this paper only pure rolling and pure sliding relative motions between the agents and the object and agent relocation are assumed. The planner allows breaking contact between agents and the object due to agent relocation. Instead of random selection (proposed in [2, 3]) we use simulated annealing (SA, [4, 8], see Appendix) algorithm at the local planner level to choose relative velocities for the agents . The purpose of using SA is to avoid local minima of the motion planning objective function which exist when there are various constraints required for defining feasible motions and forces. SA is a general-purpose optimization tool which can be easily used with any constraints in the searching space. Motion planning can be treated as an optimization problem, therefore SA is useful if

the energy function of the searching space has many minima.

A near time-optimal inspection-task-sequence planning for two cooperative industrial robots is outlined in [9]. The object to be observed is moved by one of the robots, while the other handles the inspection tool. The objective of the task-sequence planning is to find a series of near time-optimal final configurations for two arms where the inspection operations are undertaken for segment (link) motions and find a near time-optimal task sequence of inspection points. The task-sequence planning is formulated as a variation of the traveling salesman problem (see [8]), and SA is used to find a near time-optimal route. In [10] an efficient path planning approach is presented in which a path is represented by a series of via points connected by elastic strings that are subject to displacement due to collisions with obstacles as well as constraints pertaining to the domain to which path planning is applied. Obstacle regions are represented by a potential field. SA approach is used for local minima problems from the potential field. The potential field is generated by a multilayered neural network which implements SA with the temperature T of its activation functions. In [11] a robot hand manipulation algorithm is presented. Evolutionary algorithm is applied to determine when and where the fingers are moved and selects the finger of which contact point is to be changed.

The objective of this paper is to present a SA based motion planner which uses a proposed relative velocity matrix to define the motion sequence of the object and the agents. The organization of the paper is as follows: in section 2 we give the background of the problem. In section 3 the overview of the planning system is presented, section 4 describes the use of SA with the relative velocity matrix. In section 5 simulation results are presented.

2 Background of the kinematic modeling and constraints

The relative motion of two contacting rigid and piecewise smooth bodies is solved by [5] by describing the motion of the contact point on the bodies' surface. The relative velocities between the contacting bodies, and the description of the contacting surfaces are required to solve the equations of the contact motion. The main contribution of this paper is a definition of the relative velocity generating method for the kinematic model while the motion of the object and the agents is constrained during the manipulation. The motion constraints are defined in the following:

Maintaining contact between the object and the agents - it is given by the solution of the differential equations of contact motion (Montana equations).

Equilibrium condition - in case of quasi-static motion the resultant force applied on the object has to be zero. We define any non-grasping forces applied on the object as "external force". The grasp transformation is given as $G \in R^{6\times 6n}$. G transforms forces and torques between the frames of contact points $(C_1, C_2, \cdots, C_n)$ to the object frame in case of point contact model. It is used to determine the resultant force applied on the object by the agents. The equilibrium constraint is written as:

$$G[{f_1}^T, {f_2}^T, \cdots, {f_n}^T]^T = -f_{O,ext} \qquad (1)$$

$f_1, f_2, \cdots, f_n$ denote the contact forces at the contact points' frames $1, 2, \cdots, n$, for instance in 2D $f_i = f_{ix}, f_{iy})$; $f_{O,ext}$ is the resultant of the external forces applied on the object (for example gravity and/or the friction force $\mu_O mg$ between the object and the plane of motion).

Rolling constraint - in case of the relative motion between the agents and the object is rolling or stationary the relative linear (v_i) and angular velocities (ω_i) are: $v_{ix} = v_{iy} = v_{iz} = \omega_{iz} = 0$ where z is the direction of the surface normal. The contact model is point contact with friction, μ is the friction coefficient. The contact force has to be outside of the cone of friction. In our algorithm this constraint is linearized: instead of cone of friction we use pyramid of friction.

$$|f_{ix}| < -\mu f_{iz}/\sqrt{2}, \quad |f_{iy}| < -\mu f_{iz}/\sqrt{2}, \quad f_{iz} < 0$$

Sliding constraint - In case of the relative motion is pure sliding $(\omega_{ix}, \omega_{iy}, \omega_{iz} = 0)$ the contact force has to be outside of the pyramid of friction:

$$|f_{ix}| > -\mu f_{iz}, \quad |f_{iy}| > -\mu f_{iz}, \quad f_{iz} < 0$$

The tangential component of the grasping force at a given contact point should lie in the plane given by v_{ix} and v_{iy} and has to have the same direction and sign (γ is arbitrary):

$$f_{ix} = \gamma v_{ix}, \quad f_{iy} = \gamma v_{iy}, \quad \gamma > 0$$

No-collision between the agents (collision detection algorithm is implemented)

Workspace condition - For example in case of fingers (dextrous hand) the contact points have to be in the reachable space of the fingers.

Agent relocation condition - the relocating agent does not apply any contact force on the surface of the object during breaking contact and relocation.

$$f_{ix} = f_{iy} = f_{iz} = 0$$

3 Overview of the multi-agent planning system

The planner algorithm consists of two main parts: the global and the local planner (fig. 1). The task of the global planner is to search for the nominal path in the configuration space among static obstacles between the initial and the desired position and orientation of the object. A graph search algorithm is used to generate points (subgoals) in the object's quantized configuration space, the local planner tries to find admissible motion of the agents in contact with the object between the subgoals.

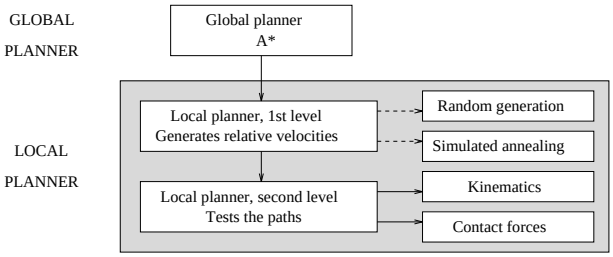

Figure 1: *Planning flowchart of the system. The global level plans the object's motion, the local planner's task is to generate relative velocities (using simulated annealing) and grasping forces for the agents.*

The constrained trajectories of the agents' motion (path) between two subgoals are generated by the local planner. The local planner includes three main parts: a) the kinematics solution (computes the trajectories), b) the force computing algorithm (solves for the contact forces) and c) the generation of the relative velocities between the agents and the object. The motion of agents is dependent on the relative velocities between the agents and the object chosen by the local planner algorithm. In the proposed method SA algorithm is used to select the relative constrained velocities and breaking contact with the object. Breaking contact for agent relocation is an additional motion to rolling and sliding contact motion in our method which is applied for example if at a given configuration (sub-goal) the contact forces are large compare to other configurations.

For object relocation an additional (auxillary) agent is introduced. The algorithm searches for a new contact point for the additional agent. Meanwhile one of the other agents breaks the contact with the object. The later agent will not participate in the grasp until another agent breaks contact. The new position (subgoal) of the relocated agent on the object is generated by a relocation algorithm. The relocation algorithm searches for the optimal new contact position for the agent. The objective funtion of relocation is defined as the sum of contact forces applied on the object by all the agents.

The kinematics for relative constrained motion of agents in the local planner is given by the solution of the differential equations of contact motion given the relative velocities. The forces at the contact points are resolved by the solution of a linear program (LP). The aim of this task is to comply with the constraints and minimize the contact forces. The constraints of the LP task are the motion constraints (described in the previous section): the equilibrium constraint and the rolling and sliding constraints (depends on the relative motion of the agent, described in section 2). In the LP task we minimize the sum of the normal component of the contact forces subject to the motion constraints (f_{i_z} denotes the normal component of the contact force between the object and the agent number i):

$$\min_f \sum_i f_{i_z} \tag{2}$$

Any solution of forces that satisfies the constraints is appropriate because the object's movement is already defined by the constraints and the geometry. The advantage of minimizing the normal component of the contact forces is to save in total energy and to spare the object if it is delicate.

4 Simulated annealing algorithm as a local planner

In our approach the energy function of the SA algorithm can be a combination of analytical (for example: equilibrium and no collision) and heuristic expressions (for example: small contact forces). For example, if the energy function is defined as the number of collisions during the motion then minimizing the energy function will minimize the number of collisions.

Consider the complete motion sequence of the object and the agents between two subgoals. The motion of the agents is parameterized by relative linear and angular velocities. In Figure (2) the first configuration and the object's position and orientation at the last configuration are given by the global planner, the in-between frames are filled up by the local planner. The object is held by three agents: the orientation of the object and agents are marked by a normal-direction radial line.

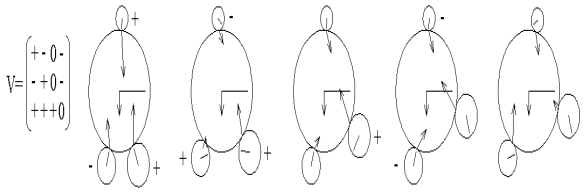

Figure 2: The motion of the object and agents, parameterized by the relative velocities (rolling clockwise and counterclockwise in this example, denoted by - and + respectively). Matrix **V** contains the velocity parameters: the rows and columns are describing the agents and velocity configurations respectively

Using SA algorithm we search for appropriate relative velocity parameters. The initial configuration of the agents and the object and the desired configuration of the object are given as subgoals by the global planner, we would like to compute the relative velocities to define the motion of the agents between two subgoals. The SA method choose from a large number of sequences using a relative velocity matrix Figure (3), and accept the best of those (which is near optimal).

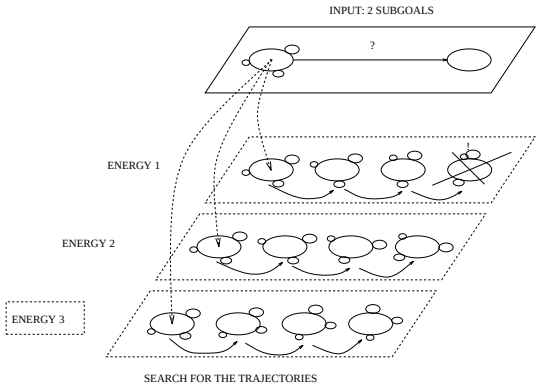

Figure 3: *Example of generation of the velocity matrix* **V**. *The local planner chooses from different motion sequences which were generated from different velocity matrices. The selection is based on the magnitudes of the energy function.*

For example let interval between two subgoals (t_s) be divided into n sub-intervals ($\Delta t = t_s/n$). For every Δt interval we would like to generate relative velocities for all agents. Consider the complete motion, there are $n*k$ parameters, where k is the number of agents, hence the complete motion can be parameterized by a $n*k$ size matrix of the relative velocities denoted by $\mathbf{V} \in \mathrm{R}^{\mathrm{n}\times\mathrm{k}}$. The relative velocities v_{ij} are chosen automatically by the SA algorithm which tries to optimize the $n*k$ size matrix of the relative velocities considering given constraints (shown in section 2).

Let the v_{ij} variable be defined as the relative velocity of the i^{th} agent in the j^{th} interval. The values of the variables can be quantized (not necessary), for example $v_{ij} \in \{0,$ *Relocation*, $v_{x_{max}}, -v_{x_{max}}, v_{y_{max}}, -v_{y_{max}}, \omega_{x_{max}}, -\omega_{x_{max}}, \omega_{y_{max}}, -\omega_{y_{max}}\}$ (Figure 2).

The energy function of the SA algorithm has several components (e.g. equation 3): the first and the second components are defined to be: a) **Force equilibrium** (applied on the object), b) **No collision** between agents. Other components can be heuristic, for example: c) **the absolute sum of contact forces** for all agents in all Δt interval has to be small. This is useful if the object is delicate or if the objective is to minimize the energy. Given a configuration of the agents and the object, equation 2 gives the minimal contact forces. The SA algorithm in this case chooses from different configurations, preferring that configuration which has the minimal absolute sum of contact forces. Another component can d) **minimize the number of relative velocity changes**. An example of the energy function can be written as:

$$E(\mathbf{V}) = K_c \sum_{j=1}^{n} c_j + K_e \sum_{j=1}^{n} e_j + K_f \sum_{i=1}^{k} \sum_{j=1}^{n} |f_{ij}| + \\ + K_g \sum_{i=1} g_i \qquad (3)$$

The first sum of (3) is the penalty due to collision (the number of collisions multiplied by a K_c constant), the second is the penalty due to no force equilibrium. The third part is the absolute sum of the contact forces, the fourth is the number of velocity changes of the agents. The definitions of the variables and constants are:

E is the energy function we want to minimize, **V** contains the relative velocities, the variables of the system of the agents (parameters), K_c, K_e, K_f, K_g are positive weight constants for collision, equilibrium, forces and velocity changes respectively; n and k denote number of intervals between two subgoals and the number of agents respectively.

$$c_j = \begin{cases} 0 & \text{if there is no collision in the j}^{\text{th}} \text{ interval} \\ 1 & \text{otherwise} \end{cases}$$

$$e_j = \begin{cases} 0 & \text{if there is equilibrium in the j}^{\text{th}} \text{ interval} \\ 1 & \text{otherwise} \end{cases}$$

f_{ij} is the contact force applied by the i^{th} agent at the j^{th} interval. g_i is the number of relative velocity change of the i^{th} agent between the two subgoals.

$K_c \approx K_e, \quad K_c, K_e \gg K_f, K_g$ because no collision and equilibrium are more important than small contact forces and small number of velocity changes.

The force component of the energy function uses equation 2. This force computing sub-algorithm (LP) is called for every configuration in every iteration. The algorithm prefers those motion sequences which have the smaller sum of contact forces, less number of collisions and velocity changes and more configurations with equilibrium.

The pseudo code of the relative velocity computing algorithm is shown in code 1.

```
1 Get 2 subgoals (start,goal)
#Two subgoals are given by the global planner
2 Generate random relative velocities(V)
#n*k initial relative velocities (n intervals, k fingers)
3 E_old = Compute-energy(V)
#Compute the complete motion and the energy for
#the complete path between the two subgoals with the
#initial values of the relative velocities.
4 WHILE (T > T_end) DO
#T is the temperature. Do the cycle until the
#desired final temperature T_end is reached.
5 Select (i, j) randomly
#Choose a variable from V.
6 Generate a random velocity (x)
#For example x = v_max or x = Relocation.
7 E_new=Compute-energy(V, v_ij = x)
#Generate the new positions for all agents for all the
#intervals and compute the energy function with the
#new value of the selected variable v_ij = x
8 ΔC = E_new − E_old
#The change of the energy function due to the
#change of the selected variable.
9 IF E_new < E_old THEN
10      LET v_ij = x, E_old = E_new
#If the change is positive then it is accepted.
11 ELSE IF e^(−ΔC/T) > Random(1) THEN
#Random(1) is a random number between 0 and 1.
12      LET v_ij := x
13 Decrease(T)
#According to the temperature schedule (see Appendix)
14 END WHILE
```

Code 1. *The SA based relative velocity generation*

In every cycle of the SA algorithm the path between the two subgoals has to be generated with the selected relative velocities then the energy function has to be computed. The number of parameters (relative velocities) is directly proportional to the number of agents and the number of intervals between two subgoals. Figure (4) shows an example for such relative motions.

The relocation algorithm scans the local coordinates of the object (Gauss frame), for each potential agent position the sum of contact forces applied by all the agents is computed as if the relocating agent would be at the potential contact point. The potential contact point with the smallest relocation function value is selected as the new contact point for the relocating agent. In Figure (5) an example for the relocation objective function is presented. Two contact point is given on the object, the sum of the contact forces (f) are shown as a function of the local coordinates (u,v) of the third contact point.

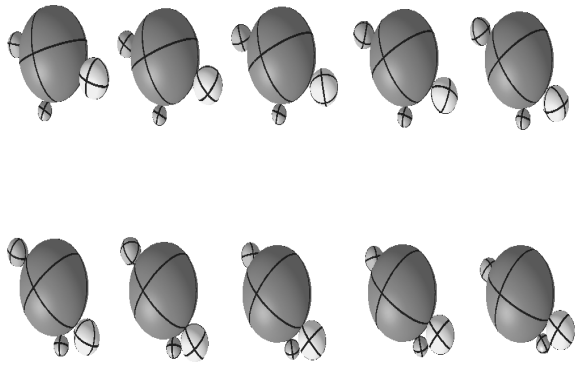

Figure 4: An example of the motion of the agents and the object

5 Discussion

The nature of the energy function highly depends on the constraints. If there is no local minimum then simply downhill search can be applied instead of SA algorithm which is equivalent to applying a SA algorithm with very small temperature. In most of the cases there are many local minima, which can be shown by executing downhill searches from different random generated initial **V** matrices. In all of our simulations the energy function had local minima which further justified the advantages of using SA.

6 Conclusion and future works

The object re-configuration problem is to find appropriate trajectories for the agents to move the object to a desired position and orientation. The motion of the agents relative to the object can be rolling, sliding, and breaking contact due to agent relocation. Our contribution is a SA based relative velocity generator of the motion planner algorithm at the local planner level. Using a relative velocity matrix (which defines the motion sequence) with a SA based algorithm the motion can be optimized by various fac-

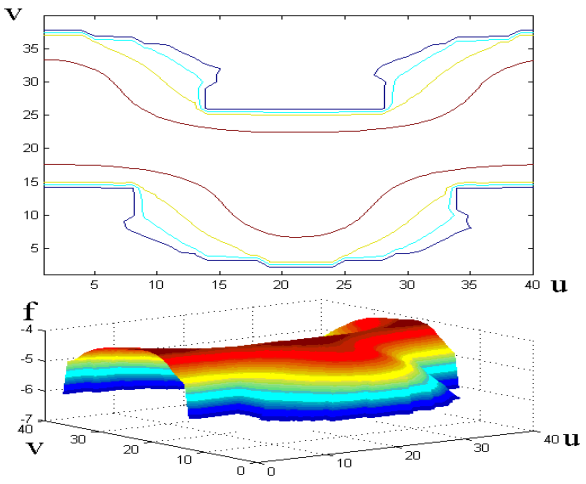

Figure 5: Example for the objective function of the relocation algorithm. Here two contact points are given. The sum of all contact points is represented by f (the objective functioon) with the corresponding u and v coordinates of the third contact point on the local coordinate frame of the object.

tors for example small contact forces. Computing trajectories for the agents with the SA based algorithm effects near-optimal paths. The method has a number of parameters which need to be fine tuned. The planner will be extended to deal with static obstacles as pseudo-agents. Using pseudo-agents more difficult motion planning tasks can be solved.

7 Appendix: Simulated annealing

Simulated annealing is a stochastic optimization technique used to avoid local minima of an objective function. Consider a system which consists of many variables and an objective function on these variables. Let the state of the system be defined by the value of all the variables. The desired state of the system has the minimal value of the objective function. Given an initial state of a system the value of the objective function (or energy function) is calculated. We attempt to move the system by changing the value of a variable searching for the desired state by small steps (iterations). In each step a variable is chosen randomly and its value is selected to be changed. A move of the system to a new state is performed with the following conditions: In case the value of the objective function of the new state is below the previous state's then the move is accepted. In case the value of the objective function of the new state is higher then the move is accepted with the probability of $e^{-\Delta C/T}$, where ΔC is the change of the objective function due to the move and T is a variable called "temperature".

The initial value of temperature is high and it is slowly decreased In high temperature (the T parameter is high) the probability of increasing the energy function is bigger than in low temperature. A good, but slow annealing schedule (the change of T in every step r, where c is constant) is:

$$T(r) \geq \frac{c}{\log(1+r)} \tag{4}$$

In the other case (energy increasing change was selected) the change of the state is accepted with a probability of $e^{-\Delta C/T}$, where ΔC is the change of the energy due to the change of the variable and T is the temperature. In the beginning T is big, so increasing the energy is accepted with a big probability. T decreases in time according to a schedule for example: T is multiplied by 0.95 in each step. After a long time the value of the energy function will be decreased almost all the time because the probability of accepting an energy increasing motion becomes very small.

ACKNOWLEDGMENT

Support for the research is provided by the Natural Sciences and Engineering Research Council of CANADA (NSERC) grant No. 611205 and the Hungarian National Research Programs under grant No. FKFP 0417/1997 and OTKA T 029072.

References

[1] K. B. Shimoga, "Robot grasp synthesis algorithms: A survey", *IJRR* , 15 (3), pp 230-266, 1996.

[2] M. Cherif and K. K. Gupta, "Planning quasi-static motions for re-configuring objects with a multi-fingered robotic hand", *IEEE ICRA* , pp 986-991, 1997.

[3] G. Vass, S. Payandeh, B. Lantos, "On Controlled Manipulation of Objects Within Multiple Dexterous Agents", *Tenth world congress on the theory of machines and mechanisms,* Vol 3, pp 1103-1108, 1999.

[4] G. Vass, S. Payandeh, B. Lantos, "Application of Simulated Annealing for Object Manipulation with Multiple Agents", *Proceedings of the Second IASTED International Conference, Control and Applications*, pp 566-571, 1999.

[5] D. Montana, "The kinematics of contact and grasp", *IJRR,* 7 (3), pp 17-31, 1988.

[6] J. Ponce, S. Sullivan, A. Sudsang, J-D. Boissonnat, J-P. Merlet, "On computing four-finger equilibrium and

force-closure grasps of polyhedral objects", *IJRR,* 16 (1), pp 11-35, Feb. 1997.

[7] M. Buss, L. Faybusovich, J. B. Moore, "Dikin-type algorithms for dextrous grasping force optimization", *IJRR,* 17 (8), pp 831-839, Aug. 1998.

[8] N. Ansari and E. Hou, *Computational intelligence for optimization, Kluwer Academic Publishers.* , pp 47-82, 1997.

[9] B. Cao, G.I. Dodds, G.W. Irwin, "A practical approach to near time-optimal inspection-task-sequence planning for two cooperative industrial robot arms", *International Journal of Robotics Research.* , Vol: 17 Iss: 8 pp 858-67, 1998.

[10] S. Lee, G. Kardaras, "Collision-free path planning with neural networks", *Proceedings, IEEE International Conference on Robotics and Automation.* , pp 3565-70 vol.4, 1997.

[11] T. Fukuda, K. Mase, Y. Hasegawa, "Robot hand manipulation by evolutionary programming", *Proceedings, IEEE International Conference on Robotics and Automation.* , pp 2458-63 vol.4, 1997.