# Asynchronous-Teams Based Collision Avoidance in PAWS

**Ju-Hsien Kao, James S. Hemmerle, Friedrich B. Prinz**

**EDRC 05-95-95**

# Asynchronous-Teams Based Collision Avoidance in PAWS

Ju-Hsien Kao, James S. Hemmerle, Friedrich B. Prinz

Engineering Design Research Center
Carnegie Mellon University
Pittsburgh PA 15213

1 June 1995

## Abstract

This paper describes the Asynchronous Teams (A-Teams) based collision avoidance algorithm in the Programmable Automated Welding System's Off-line Programming (PAWS-OLP) system. A-Teams organize different software or operators in an asynchronous way so that a variety of simple or sophisticated agents can cooperate to produce better results in a global sense. Joint limits, minimal joint motion, smooth joint motion, and task quality are also considered in this collision avoidance algorithm. An example, using a six degree of freedom robot PUMA762, is investigated.

## 1. Introduction

Due to the additional degrees of freedom, kinematically redundant manipulators make a system more flexible in achieving task requirements. Several research papers have addressed the resolution of redundancy (i.e. the selection of a set of joint values from the infinite number of possibilities). Pseudo-Inverse approaches give the minimal norm solutions together with homogenous solutions under the constraint of the specified end-effector velocity([15], [16], [18], [19], [20]). Extended Jacobian methods add additional constraints to the original problem so that the resulting Jacobians now are invertible([21], [22], [23], [24]). Both of these methods use local information only([7]), meaning only information at the specified point is considered. Also, they require well defined gradients for the criterion functions they add.

Collision avoidance usually is a necessary constraint for manufacturing tasks. Some papers can been found to apply this constraint in redundancy resolution^11], [12], [13], [15]). Most of the work deals with simple geometry (lines, triangles, or rectangles) in two dimensional space for planar manipulators. The objective is to enlarge the distance between two nearest points to ensure safety. Distance between objects usually can be found analytically in these cases. Other objectives which have been addressed for redundancy resolution are singularity avoidance, joint limits constraint, and other measures of dexterity. Once objectives or constraints are defined, the resolution process can be easily converted to an optimization problem.

In the PAWS-OLP system, a numerical scheme based on gradient methods was first implemented to solve this optimization problem. But in order to make this gradient-based approach applicable, the following conditions must be established: first, all the points in the initial path must be inside the robot envelope so that the inverse kinematics solutions are available. Second, the initial path must be collision free. Otherwise, gradients for moving objects toward safer positions are unavailable. Third, the initiafpath needs to be close to the global optimum to avoid being trapped in the local minimum. Unfortunately, there's no guarantee that such an initial path can be found.

Due to the difficulties in meeting these criteria, the Asynchronous Team (A-Team) was used to generate an initial path that is collision free and is inside the work envelope as an input for the numerical optimization scheme. Since the numerical scheme requires a great number of calculations of distances between objects in the environment in order to keep the path collision free, and it is usually hard to incorporate multiple objectives into a single measurement for the

This work is sponsored by the Navy Joint Center (NJC) and the Engineering Design Research Center (EDRC) at Carnegie Mellon University

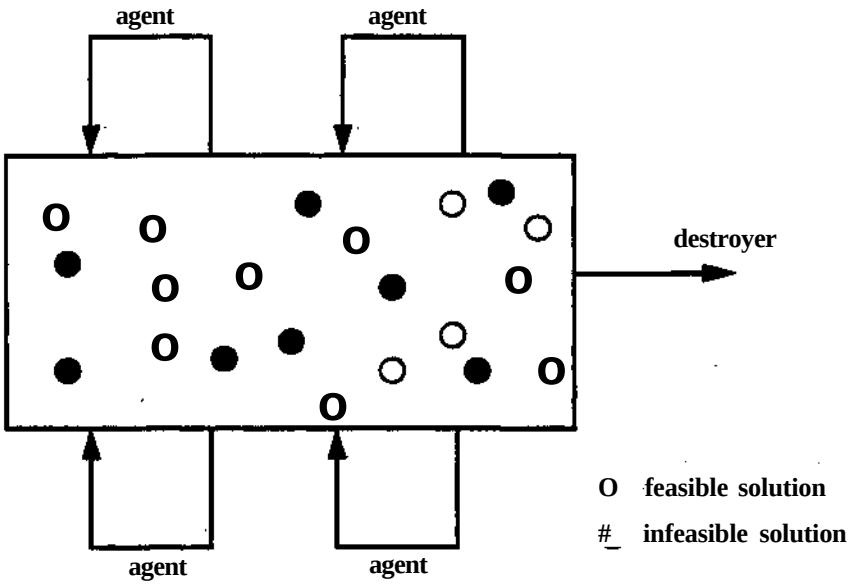



**Fig.l. An A-Team diagram in PAWS-OLP**

numerical scheme, the A-Team methodology was investigated for solving the entire path planning problem in the PAWS-OLP system. This algorithm is appropriate for the following reasons: first, the PAWS-OLP system is an off-line motion planner, so time is not of great concerned. Secondly, a gradient-based approach can be one of the agents in A-Teams if gradients are available. Heuristic based operators can also be used as agents when gradient based approaches don't work. Third, A-Teanis allow operators to work on different subsets of the problem, which reduces its complexity. Fourth, since A-Teams work on population of solutions, local minimum problems may be avoided. Finally, multi-objective problems can be handled better than in numerical schemes.

# 2. Asynchronous Teams

An Asynchronous Team (A-Team) is a solution algorithm which encapsulates different software operators, or even existing algorithms, into autonomous asynchronous agents. Through cooperation, complex problems, which individual algorithms could not resolve, can be solved. A-Teams were proposed by Sarosh Talukdar in 1990. He defined an A-Team ([1], [2], [4], [6]):

> *to be any super object whose agents are autonomous (so new ones can be added without needing any managerial super structure), whose communications are asynchronous (so all its agents can work in parallel all the time) and whose dataflow is cyclic (so iteration andfeedback are possible).*

In general, an agent, which encapsulates a solution algorithm, reads a solution, modifies it according to its algorithm to produce a new solution, and insert this new solution into the population. All agents are autonomous and no supervisor controls the sequence of the agents. When adding or deleting an agent, there is no need to notify the supervisor (since no supervisor exists) or the other agents. When adding or deleting an objective or constraint, no agents need to be modified. Instead, some agents related to the objective are added or subtracted. Since each agent can focus on one or more objectives or constraints, the problem can be decomposed into subproblems which can be addressed

by simpler means.

The communication between agents is asynchronous through the shared memory. Each agent is the supervisor of itself and has its own scheduler. They can work in parallel without delaying others, since agents do not control each other. All the agents work on the populations of solutions and need not to wait for the other agents' work. Since all communications are via the populations and the data flow is cyclic, cooperation is possible.

The population of solutions may have different representations and could be stored in different memories according to the application and the algorithms being encapsulated. Agents also have their own strategies for selecting appropriate solutions from among the many available in the shared memories as their inputs and generating new solutions into the desired memories. While the process is running, more and more new solutions are added to the memories, and thus, a population-size control strategy must be used to maintain the quality of the population. A large size for memory could make the population diverse, which avoids the solutions being trapped in the local region, but make the process less efficient since agents might spend lots of time in working on unpromising solutions.

Several problems have been solved by A-Teams, including the traveler salesman problem (TSP) ([5]), design of 2-D RMMS (reconfigurablc modular manipulator systemsXP]), and job-shop scheduling problems. De Souza organized algorithms for solving TSP in an A-Team and the A-Team performed better than any single of these algorithm. He also found A-Teams are scale efficient: the more agents that are put in, the better it can perform. Murthy used several simple agents, each of which worked on a single objective only, in the 2-D RMMS design problem. By putting these agents in an A-Team, good designs were easily found. He also showed that multi-objective problems can be addressed and that the objectives need not be combined into a single measurement. De Souza's and Murthy's work has verified that, through the A-Teams organization, large scale problems with no single satisfactory algorithms can be successfully resolved.

In general, A-Teams are capable of resolving problems which single algorithms can't solve well. Problems which can be solved efficiently by numerical approaches (with well-defined gradients and continuous objective functions) and whose search space is smooth, are usually not appropriate for A-Teams.

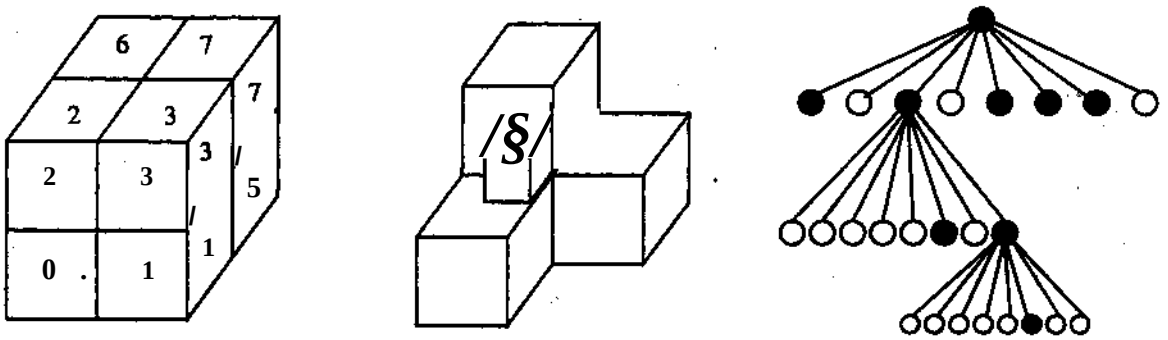
In our A-Team implementation, all solutions have the same representation and could be stored in one memory. An A-Team diagram for this trajectory planning problem in the PAWS-OLP system is shown in Fig.l.

# 3. Collision Detection and Distance Calculation Algorithm

Collision between two objects is detected by testing whether the boundary of one object is inside the volume of the other. The boundary is stored as a wire frame model (since we deal only with linear facetted objects), and the volume is represented by an octree. Thus, a dual representation of a wire frame and an octree is used.

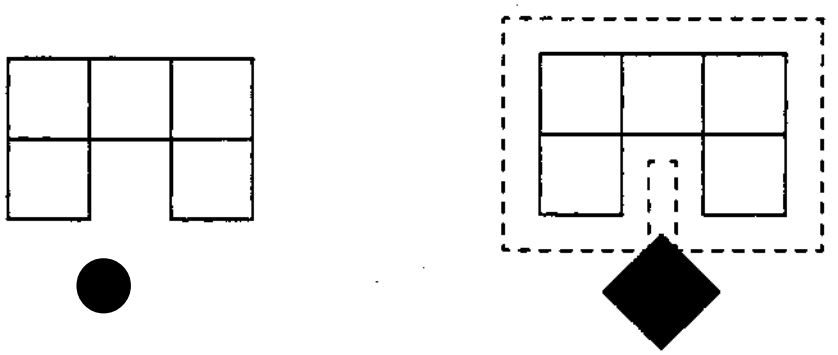
The octree of the object is generated by a recursive subdivision of the bounding box of the object into eight octants, which are stored as nodes in an 8-ary tree. If the octant is fully inside or outside the object, the nodes are terminated and termed black(full) or white(empty) respectively. Otherwise, the subdivision is continued until a desired resolution is reached. Usually, the levels of the octree are set to 6-8, for which resolution goes to 1/64-1/256 of the length of the object. (Fig. 2.)

To test for a collision between two objects, the coordinate of the object represented by a wire frame model is transformed into the frame of the other object represented by an octree. The bounding boxes of objects are first tested to check if they are overlapped. The vertices of the transformed wire frame model are then tested by traversing the octree of the other object until the octants containing the vertices are found. The edges are then tested in the same way until all the segments (which are divided by octants) are in either full or empty octants. The process stops immediately when the bounding boxes do not overlap, in which case they are clearly collision free, and when any of the ver-




labels of octants of a cube  an example object  octree of the example object

**Fig.2 An example of octree generation**



original object  obtain distance by growing object

**Fig.3 An example of distance calculation**

tices or any portion of the edges is inside the full octant, in which case they collide.

Distance between two objects is calculated by an algorithm based on the collision detection algorithm described above. The octree of one object is grown until it collides with the other. The growth amount of the octree when collision occurs then becomes the distance measure between these two objects. Finding such a critical growth amount is accomplished by a bisection method. (Fig. 3.)

This collision detection and distance calculation algorithm can be applied on any arbitrarily shaped objects (they need not be convex) and octrees need only be generated once (since generations of octrees are time consuming tasks depending on how complex the object is). The collision detection algorithm is extremely quick since it only needs a few comparisons and additions during each traversal through the octree. Objects which collide will take less time to be tested then those without collision, since the testing process will immediately stop once a collision is detected. Comparisons with other collision detection approaches can be found in [25].

In our A-Teams implementation, information about whether or not the objects interfered is used, not the distance between them. (The distance is used in the numerical algorithms.) There are three reasons for this. Normally, computation time for calculating distance between two 3D complex geometric models is much longer than that for detecting whether or not collisions occurs between two models. Additionally, when objects interfere with each other, the dis-

tance between objects is zero and the gradient used to move them into safe configurations is not available in our octree implementation. Also, as long as two objects are separated from each other by a certain distance, they are safe. There is no need to find a solution with the robot links as far away from the other objects as possible.

# 4. Implementation

## 4.1. Problem Formulation

In this section, the focus will be the PAWS-OLP system. A welding gun serves as the end effector of the robot in the system. The tip of the welding gun must follow the given seam path, which is composed of a sequence of points, in order to perform the welding tasks. No component of the system should collide with any other component or any object in the environment, and the joint angles given to the robot system must be within its joint limits. In addition to these, robot joint motion needs to be minimized to avoid jerky motion and the robot trajectories must be smooth to make it easy to track along the seam path. Also, the orientation of the tool must be as close to the optimal welding orientation as possible to obtain the good welding quality.

A kinematic diagram for robotic motion along a given path is shown in Fig.4. This figure describes the relationship among kinematics transforms in order to have the tool-tip move along the path points. The robot base transform specifies the robot base frame relative to the world frame, while the part transform specifies the part frame relative to the world frame. The robot forward transform describes the frame of the face plate of the robot manipulator with respect to the robot base frame, while the tool transform describes the frame of tool tip with respect to the face plate. The point transform is the point on a path relative to the frame of the part. The point orientation change is the transform to specify the modification of point orientation relative to its initial value. Because of the constraint that the tool tip must follow the path points, this kinematics chain forms a closed loop.

For a *m* (m≥6) DOF robot working in 3D space with positions and orientations of the trajectory specified, the system has *(m-6)* degree of redundancy. Points with specified positions and unspecified orientations will make the system have *(m-3)* degree of redundancy. An optimization algorithm can be used to find a good solution which satisfies task constraints and minimizes objectives among these infinite possible solutions. Here, the solution is the robot joint trajectories which have the tip of the welding gun move along the given seam path, and is composed of a sequence of robot joints.

A mathematical form to describe this trajectory planning problem is stated as follows:

Given: $T_R$, T^ Tp> $T_{PF}$> $T_F$(6)

**Find:** 6,

subject to TRTJKGJTT = $T_P$Tpp5Tpp(x),

**$C_i(\theta)$, i=1,2...**

minimizing Fj(0), i=l,2...

**where $T_R$: robot base transform,**

**$T_T$: robot tool transform,**

**$T_P$: part transform,**

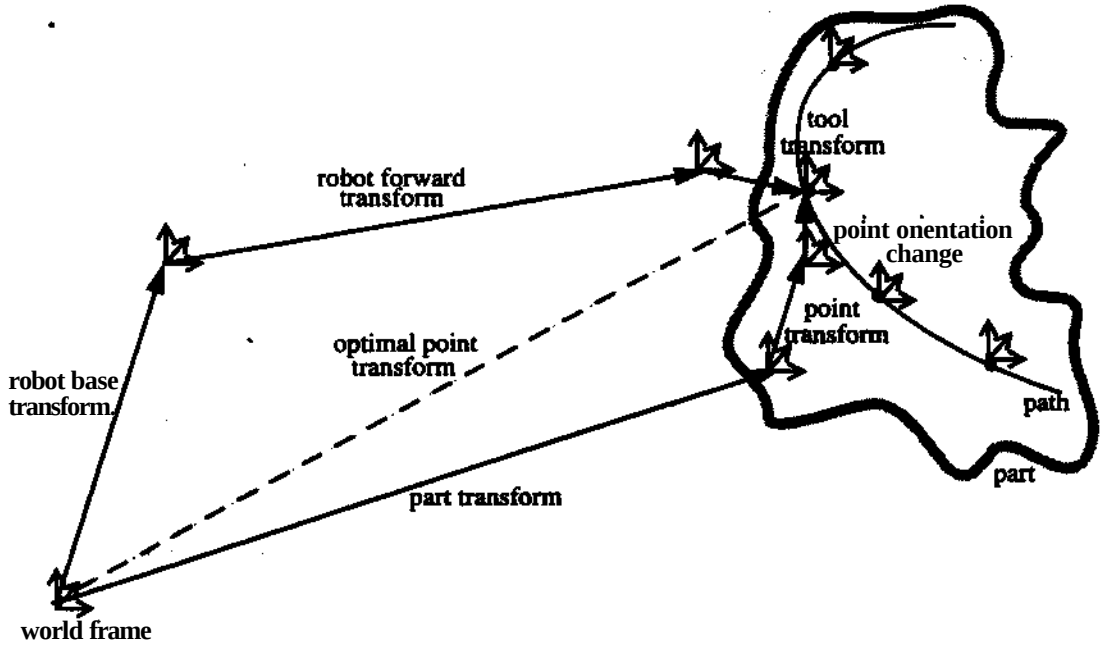



**Fig.4. Kinematic chain of a trajectory following problem for a single robot system**

**$T_{PP}$: path point transform,**

**5Tpp(x): path point transform change,**

Tp(6): **robot forward transform,**

**x: changes of the optimal welding orientations**

**6: robot joints,**

**C|(8): constraint function for constraint i,**

**$F_i$(6): objective function for objective i**

The constraints used in this trajectory planning problem are collision free path, and within-limit joint values, denoted as $C_{coUision}$ and C/£miw, respectively. Minimal joint motion (to decrease the joint velocity), smooth joint motion (to decrease the joint acceleration), and minimal changes of the optimal point orientations (to maintain good welding quality) are the objectives, and are denoted as $F_{motion}$, $F_{smooth}$ and ^*direction** respectively.

These constraint and objective functions are defined as follows:

$$Collision,\ i\ (**.\ ^{r}*\ ^{T}P>\ ^{T}V\ ^{OCtreeS}\wedge\ *^{True}<\ ^{V/}$$

$$C_{limits,\,i,j}\left(\theta_{i,j}^{max},\theta_{i,j},\theta_{i,j}^{min}\right)\neq True,\ \forall i,\forall j$$

$$E_{motion} = \sum_{i}\sum_{j} W^{j}_{motion} A_{V\ i}\left(\frac{\theta_{i,j}-\theta_{i-1,j}}{d_{i}-d_{i-1}}\right)^{2}$$

$$F_{smooth} = \sum_{i}\sum_{y} Zt_{i}Zt_{y}\wedge smoothl_{V}\left(\frac{\theta_{l+1,j}-\theta_{i,j}}{\ldots}-\frac{\theta_{i,j}-\theta_{i-1,j}}{d_{i}-d_{i-1}}\right)^{2}$$

$$\wedge direction = \frac{1}{\sum_{i} V}\left(\wedge direction,\ r_{r}^{2} + \wedge direction,\ p_{p}^{2} + \wedge direction,\ w_{w}^{2}\right)$$

where i represents the point number, $j$ represents the robot joint number. $\theta_{i,j}$ denotes the $j$th robot joint value at point i, $d_i$ denotes the traveling distance from the first point to the $i$ th point. $x_{ir}$ $x_{ip}$, $x_{iw}$ are the variables roll, pitch, yaw respectively of the change of the optimal welding orientation at point i and $\wedge dirtctwtun$ $\wedge direction$ $\wedge dinctioruw$ are their weights for the task quality objective. For symmetric tools, $W_{dir€Ctiorur}$ could be set to zero so that rotations about the tool axis have no effect on this objective. $C_{coUisUm\%i}$ will return *True* if collision occurs when the robot reaches point i. $C_{llrmtslr}$ is *True* when joint; is outside the limits when point $i$ is reached. In the case of constant speed motion of the end-effector in cartesian space, the average joint rates during one segment can be expressed as the ratio of joint changes to the traveling distance <$motion)- The change of the average joint rates thus can be defined as their joint rates difference, assuming the acceleration time is fixed ($F_{smooth}$).

## 4.2. Solution Representation in A-Teams

As stated previously, solutions in our trajectory planning problem are the robot joint trajectories. Additionally, only the orientations of path points are allowed to be changed in order to achieve the task requirement while satisfying trajectory following constraints. For a six-degree-of-freedom robot manipulator, this constraint, with a specified configuration type, such as righty or lefty, and specified orientation variables, results in an unique set of joint values, if the point is inside the robot envelope. Thus, given the configuration type of the robot motion, point orientation variables can be chosen as decision variables. Robot joints can be calculated by the inverse kinematic functions once orientation variables are determined. This reduces the number of unknown variables from $6n$ to $3n$ ($n$ is the number of path points), but needs extra care on resolving the unique set of robot joints due to the periodical characteristics of angles.

Both of point orientation variables and robot joints are stored in the solution. Overall performance and individual evaluations for different objectives or constraints are also contained in the solution representation(Fig.5).

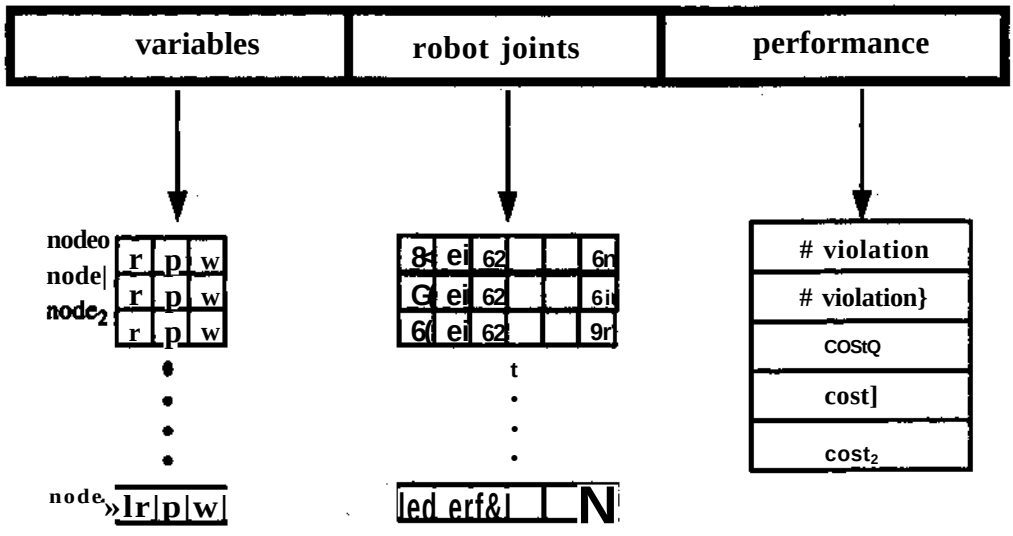



**Fig.S. Solution representation in A-Teams for collision avoidance algorithm**

## 4.3. Agents

Agents in A-Teams can be arbitrarily simple or complex. For example, they can be based on gradient techniques, blind search, or heuristic algorithms. Each agent can concentrate on one objective, multiple objectives, or do some special operations. In our implementation, all the agents select solutions as inputs from the single memory and output new solutions into that memory. The solution selection mechanism is based on Murthy's modified Means-Ends-Analysis ([3]).

A set of agents which are used in solving trajectory planning problems are introduced as follows. (5 denotes an instance of solutions in the shared memory. <S> denotes path, *p* denotes the path point. $X_s$ denotes decision variables for solution *S*. $x_{ij}$ denotes the ;' th decision variable at point *i*. $x_t$ denotes the decision variable vector at point i. 8 denotes a small number, *m* denotes an integer. 0$ denotes the set of robot joints at all points in solution *S*. 6; denotes the robot joints at point *i.)*

1. Search-Locaily-for-Collision-Avoidance-1

*for one point p, (pe* O, O e *S)and* ($C_{colUsion,\ p}$== *True),*

*try all possible sets of*$X_s$, *X$={\x* j| *if(i==p) (x\^* € {*ij+5, *y-5}, V/), *XQ* € 5}

This operator only works on collision avoidance. It attempts to find, at a certain point, an orientation for which the parts do not collide. The operator does the local search around $x_p$ based on the generate-and-test approach and can easily find such an orientation if the original orientation is close to non-collision regions.

2. **Search-Locally-for-Collision-Avoidance-2**

*for one point* p, *(p e* O, O € 5, $C_{coUisiontp}$ == True), ami one *variable of* $x_{pj}$^

*try all possible sets of* X5:

$X_S$={$x_{ij}$[1] *if(i~P andj==k)* (xy € {xy+mS, xy+fm-/^,....., Xjj-fm-/)6^y-mS}), $x_y$ € 5}

This operator also works on collision avoidance only. One decision variable of a collision point in path 0 for solution *S* is modified and tested. While a non-colliding orientation my be found for point p, the value of variable $x_{pk}$ might become too large, which could worsen other objectives a lot.

3. **Cross-Over-with-Another-Solution-For-Collision-Points**

*for point* $pi_f$ *(pi e* $<b_h$ <& € *Sj), and* ($C_{coUisiontpJ}$~ *True),*

*point p2, (p2* e <J>, $O_2$ € $S_2$), *and (CcoUisum.p2~- False), and (pi and p2 have the same position),*

*do cross over:* $Xl_s$=fxliJ *if(i==p)* (xl^xl^ V/), $xl_u$ € 5, $x2_{itj}$ € 5}

To increase efficiency, an operator similar to cross-over operators in Genetic Algorithms is used. The preconditions for this operator to improve the solutions is one point(p/J with collision in one solution(Sy) must be collision free (at the corresponding point, $p_2$) in another solution ($S_2$). Variables in p/, will be overwritten by variables in $p_2$. It reduces efforts in searching non-collision points.

4. **Mimic-Neighboring-Non-Collision-Points**

*for one point pi, (pi* € O, O € 5) *and* ($C_{coUisioru\ pI}$ == *True),*

*another point p2, (p2 e* O, O € *S),* ($C_{coi}i_{sioru\ p2}$==*False), and (Neighbors(pl, p2)==True),*

*try* $X_s$: $X_s$= $fx_{it}$/ *if(i==pl)* ($x_{pIJ}$=$x_{p2J}$,*Vj),* $x_{itj}$ € 5}

This agent copies the orientation vector from a neighboring non-collision point to a point with collisions. This reduces the effort for searching for non-colliding orientations, but does not work well when the part geometry changes abruptly around these points.

5. **Randomly-Modify-Variables-of-Collision-Points**

*for one point p, (pe* O, O € *S)and* ($C_{collisioni\ p}$ == *True),*

*set* $X_s$: $X_s$={$x_{ij}$l *if(i==p)* ($x_{ij}ftx_{hi}$+*random(),* V/;, $x_{uj}$ *e S]*

*or*

*for all points P:* P= {pip € 0>, 0> € 5, *and* $C_{collisiorup}$==*True]*

*setX$_s$:* X$_s$={Xi jl *if(ieP) (x^x^randomO,* V/), $x_{ij}e$ 5}

, where *random()* produces a random number.

To diversify the solution space, and to increase the chance to find collision-free solution elsewhere, these random modification operators serve as disturbers to force variables to jump outside their local region. Normally they are not responsible for producing non-collision solutions, but those solutions produced by these random operators increase the chances to generate feasible solutions by other operators in the future.

## 6. Avoid-Joint-Limits-Numerically

In order to guide the search, a penalty function for the within-joint-limits constraint can be defined so that joints within limits have very small value and joints outside limits have very large value (the farther the joint values are outside the limits, the larger the penalty function value is).

$$F_{limits,i}(\theta) = \sum_j \left\{ \begin{array}{l} \frac{1.}{\left(\theta^{max}_{i,j} - \theta^{min}_{i,j} - \varepsilon\right)\varepsilon} \quad if\left(\theta^{min}_{i,j} \le \theta_{i,j} \le \theta^{max}_{i,j} - \varepsilon\right) \\ 1.0 \quad otherwise \end{array} \right.$$

$$F_{limits,i}(\theta) = \sum_j \left\{ \begin{array}{l} \frac{1.0}{\left(\theta^{max}_{i,j} - \theta^{min}_{i,j} - \varepsilon\right)\varepsilon} + \frac{\left(\theta_{i,j} - \theta^{max}_{i,j}\right)}{\varepsilon^2}, \quad if\left(\theta_{i,j} > \theta^{max}_{i,j} - \varepsilon\right) \\ \frac{1.0}{\left(\theta^{max}_{i,j} - \theta^{min}_{i,j} - \varepsilon\right)\varepsilon} \quad otherwise \end{array} \right.$$

, where £ is a very small real number. The gradient of the penalty function relative to variables at one point can be easily obtained. Then a unit local gradient direction will be

$$\delta x_i = \frac{\frac{d}{dQ_i^F} limits, \; (W^{*\wedge i} - aJ_t}{\left| \frac{d}{d\theta_i} F_{limits,i}(\theta) \cdot \frac{\partial \theta_i}{\partial x_i} \right|}$$

where

$$\frac{d}{dQ_i^F} limits, / (9) = \left(-JQ\frac{d}{i,0}^F limits, \; i\,(^\circ) \longrightarrow jQ\frac{d}{i,n-1}^F limits, \quad i\,(\theta)\right)$$

where *n* is the number of decision variables for one point. An operator for modifying the decision variables toward within-joint-limits status can be written as

*for one point p, (p* € 0 , 0 € 5)

*iteratively try X$_s$: X$_s$=[x$_{(}$J if(i~~p~~) ($x_{ij}$=$x_{itj}$-scale-&$c_{ij}$), $x_{UJ}e$ S]*

This gradient-based operator works on the within-joint-limits constraint. The penalty function is only used in this operator to guide variables toward feasible regions.

**7. Reduce- Variable- Values-for-Better-Task-Quality**

*for one point p, (p e* O, O € *S)*

*setX$_s$: X$_s$=[Xij\ if(i==p)* K y = - V *scale • probix$_Q$))$_t$ x$_{Uj}$ e S]*

**,where** *prvbfxQ)* **returns a probability value which depends on the magnitude of jcy.** *x$_{ij}$j* **needs to be controlled small in order to maintain the welding quality.**

**8. Reduce-Joint-Motion**

**This operator is based on gradients of the objective function for minimal joint motion. With local gradient directions available, variables could be modified toward the minimal joint motion.**

*for one point p, (p* e O, O e 5)

*try X$_s$:* X$_5$= *{x$_{Uj}$\ if(i==p) (x$_{iJ}$=x$_{iJ}$-scale-&x$_{ij}$), x$_{tJ}$* e *S]*

*or*

*for all points P: P={p\p* € O, Oe 5}

*try* X$_s$: X$_s$={Xijl *if(ieP) (x$_i$^x$_{i_1}$^scale'8x$_{ip}$, Vj), x$_{ij}$e S]*

**, where** *Sx$_{ij}$* **is they' th element of the unit gradient direction for the objective function of minimal joint motion at point i.**

**The purpose of this operator is not to find the local minimum for this objective, which could cause some constraints to be violated. Instead, it takes a step in that direction to improve the solutions.**

**9. Smooth-Joint-Motion**

**This operator works in the similar way with Reduce-Joint-Motion operator. It smooths joint motion of solutions by taking the gradients of the smooth-joint-motion objective function.**

**10. Unwind-for-Joint-Motion**

*for all points P:* P= *(p\p* € 0 , 0 € *S]*

*try* 0$_S$: 0$_S$= {Oyl *if(i e P) (Q^Qtf m-2n,* Vy), x$_y$ € 5}

**, where** *m* **is selected so that** *Q$_{\dot{x}}$*j **is closest to** *Q^$_{i}$*j **in a period of 2rc.**

**The end effector's position and orientation will be the same if one joint value is changed from 0 to 9+271. This operator changes joint values by multiples of** *2n* **to eliminate unnecessary turns for joints.**

**11. Randomly-Modify-Variables**

**This operator is similar with Randomly-Modify-Variables-of-Collision-Points operator. But this one works on any point, not only collision points. It is used to diversify the solution space. It worsens solutions in general, but gives the chance to find out better solutions by other operators in the future.**

*for one point p, (p* e *<&, 4>€ 5)*

*set* $X_s$: Xg={xyl *if(i==p) (x^x^random(),* V/), *XQ e S]*

**12. Destroyer**

**This agent deletes unpromising solutions from the population of solutions to control the size of memory. It deletes the infeasible solutions first, if any, then feasible but dominated solutions, then feasible and non-dominated solutions. Infeasible solutions are those who have constraint violations, and solutions without any constraint violation are called feasible solutions. A solution is called non-dominated if no other solutions have better evaluation values in every objective and constraint. Among infeasible solutions, those who have larger number of constraint violations are deleted first. Otherwise, those who have higher weighted cost function values have higher priorities to be deleted. The process continues until the desired size of memory is reached.**

**13-18. Others**

**In our A-Teams implementation, there are 6 other agents not mentioned here which are basically subsets of the agents mentioned above.**

# 5. Results

**As an example of the method, the trajectory of a six-DOF robot PUMA welding along the corner of a rectangular with-cover box(Fig.6) is examined.**

**For the initial path points, the positions are along the edges and the orientations are given by the optimal welding orientations, whose normal directions are the bisections of the faces forming the edges. The part position and orientation and the robot configuration are given as inputs. The A-Teams contained 18 agents running for this planning problem. A destroyer based on the weighted cost function is used to take bad candidate solutions out of the shared memory. The size of the population in memory is controlled to be 50. The optimization is stopped when all agents have worked on all of the solutions in the memory and couldn't produce any better solutions than those in the memory.**

**Twenty points are used to define the welding path in this example. The most difficult locations for the robot to track the desired trajectory are around the corners. It is likely that robot links interfere with the covers of the box although the welding torch is clearly collision-free.**

**Initially, there are 13 points at which the robot collides with the part, and 7 points have joint values outside the limits. After a few seconds of running, the A-Team produces the first feasible solution in the solution memory. As the process keeps running, solutions in memory get better and better. The process stops when no agents can improve the solutions and the solution with the best performance in the shared memory is outputted.**

Because of random operation and probabilities involved, the A-Team does not necessarily produce the same results for different runs. In average, it takes about SI.32 seconds to find the first feasible solution and about 7 minutes to get stalled with the final population of solutions on an DEC alpha machine in this example.

Fig.7 shows the robot joint trajectories in the initial solution, the first feasible solution (no collision, no limits violation) produced by agents, and the final best solution in the memory respectively. In the initial solution, joint values around corners change suddenly, and are outside limits from time to time. The first feasible solution, though containing no collision or limits violations, requires large joint motions from point to point Finally, the A-Team finds a fairly good solution. A comparison of simulated result for the initial solution and the final result is shown in Fig.8.

# 6. Conclusions

A-Teams are capable of overcoming drawbacks of other approaches for redundancy resolution. It works in a general way so that the geometry of the objects in the environment can be arbitrary. Any redundant robot manipulator can be used as long as their inverse kinematics is provided. Other objectives and the modification strategies for each objective can be easily added to or subtracted from the A-Teams, without changing any other agents. By keeping the diversity of solutions in the population and making autonomous agents work iteratively and asynchronously, A-Teams can produce fairly good solutions in the global sense.

Future work on this collision avoidance algorithm will focus on the inclusion of singularity avoidance and part placement.

# Reference

1. P.S. de Souza and S.N. Talukdar, "Asynchronous Organizations for Multi-Algorithm Problems/* ACM Symposium on Applied Computing, Indianapolis, IN, February, 1993.

2. S.N. Talukdar and P.S. de Souza, "Asynchronous Teams," Second SIAM Conf. on Linear Algebra: Signals, Systems, and Control, San Francisco, CA, Nov., 1990.

3. S. Murthy, "Synergy in cooperating agents: designing manipulators from task specifications/" Ph.D. dissertation. Electrical and Computer Engineering Department, Carnegie Mellon University, Pittsburgh, PA, 1992.

4. S.N.Talukdar, P.S. de Souza, R. Quadrel, and V.C. Ramesh, "A-Teams: Multi-agent Organizations for Distributed Iteration," Proceedings of the EPRI/NSF Workshop on Application of Advanced Mathematics to Power Systems, Redwood City, CA Sept., 1991.

5. P.S. de Souza, "Asynchronous Organizations for Multi-Algorithm Problems," Ph.D. dissertation, Electrical and Computer Engineering Department, Carnegie Mellon University, Pittsburgh, PA, 1993.

6. S.N. Talukdar, R. Quadrel, R. Christie, and V.C. Ramesh, "Multi-agent Organizations for Real-Time Operations," Proceedings of the IEEE, vol.80, no.5, May 1992.

7. J.S. Hemmerle, "Optimal Path Placement for Kinematically Redundant Manipulators", Ph.D. dissertation, Mechanical Engineering Department, Carnegie Mellon University, Pittsburgh, PA, 1989.

8. J.S. Hemmerle, M.Terk, E.L. Gursoz, F.B. Prinz, T.E. Doyle, "Next Generation Manufacturing Task Planner for Robotic Arc Welding", ISA Transactions, vol.31, no.2, 1992.

**9. P. Wenger, P. Chedmail, F. Reynier, [44]A Global Analysis of Following Trajectories by Redundant Manipulators in the Presence of Obstacles", Proceedings of IEEE Robotic and Automation, 1993.**

**10. C. W. Warren, "Global Path Planning Using Artificial Potential Fields", Proceedings of IEEE Robotic and Automation, 1989.**

**11. J. Baillieul, "Avoiding Obstacles and Resolving Kinematic Redundancy", IEEE Int'l Conf. on Rob. and Auto., pp 1698-1704.**

**12. F. Cheng, T. Chen, Y. Wang, Y Sun, "Obstacle Avoidance for Redundant Manipulators Using the Compact QP Method", IEEE Int'l Conf. on Rob. and Auto., pp 262-69,1993.**

**13. P. Wenger, P. Chedmail, F. Reynier, "A Global Analysis of Following Trajectories by Redundant Manipulators in the Presence of Obstacles", IEEE Int'l Conf. on Rob. and Auto., pp 901-06, 1993.**

**14. A.A. Maciejewski, J.M. Reagin, "A Parallel Algorithm and Architecture for the Control of Kinematically Redundant Manipulators", IEEE Transactions on Robotics and Automation, Vol. 10, No.4, pp 405-414, August 1994.**

**15. A.A. Maciejewski, C.A. Klein, "Obstacle Avoidance for Kinematically Redundant Manipulators in Dynamically Varying Environments", The International Journal of Robotics Research, Vol.4, No.3, Fall 1985, pp 109-117.**

**16. K.R. Hareendra Varma, M.Z. Huang, "Geometric-Based Efficient Rate Coordination for Manipulators with Kinematic Redundancy", Journal of Robotic Systems, Vol.14, No.4, pp 275-285, 1995.**

**17. Y. Nakamura, "Advanced Robotics: Redundancy and Optimization", Addison-Wesley Publishing Company, Inc., 1991.**

**18. D.E. Whitney, "Resolved motion rate control of manipulators and human prostheses", IEEE Trans. Man-Machine Systems, MMS-10, No.2, pp 47-53, 1969.**

**19. C. Klein, C. Huang, "Review of Pseudoinverse Control for Use with Kinematically Redundant Manipulators", IEEE Trans, on Systems, Man, and Cybernetics, vol. SMC-13, No. 3, pp 245-50, March/April 1983.**

**20. C. A. Klein, "Use of Redundancy in the Design of Robotic Systems", Rob. Res. - The 2nd Int'l Symp. pp 207-14, 1985.**

**21. J. Baillieul, "A Constraint Oriented Approach to Inverse Problems for Kinematically Redundant Manipulators", IEEE International Conference on Robotics and Automation, pp 1827-1833, Raleigh, NC, 1987.**

**22. Y. Gu, "Dynamics and Control for Redundant Robots", IEEE Int'l Conf. on Rob. and Auto., pp 195-99, 1988.**

**23. J. Luh, Y Gu, "Industrial Robots with Seven Joints", IEEE Int'l Conf. on Rob. and Auto., pp 1010-15, 1985.**

**24. M. Vukobratovic, M. Kircanski, "A Method for Optimal Synthesis of Manipulation Robot Trajectories", Trans, of the ASME J. of Dynamic Systems, Measurement, and Control, vol. 104, pp 188-93, June 1982.**

**25. J. S. Hemmerle, E. L. Gursoz, F. B. Prinz, "Rapid Interference Detection", NATO ASI Series, Vol. F71, Expert Systems and Robotics, pp 233-41, 1991.**

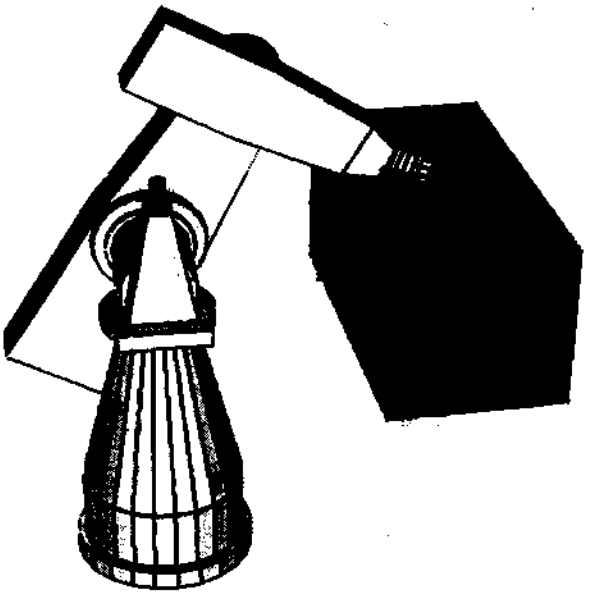


**Fig.6. PUMA762 is welding along corners of a box.**

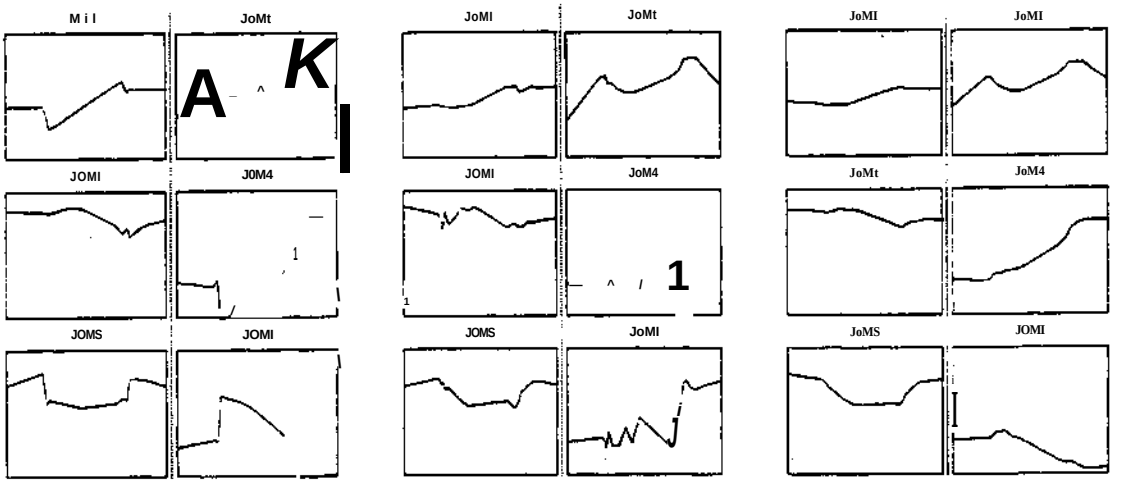
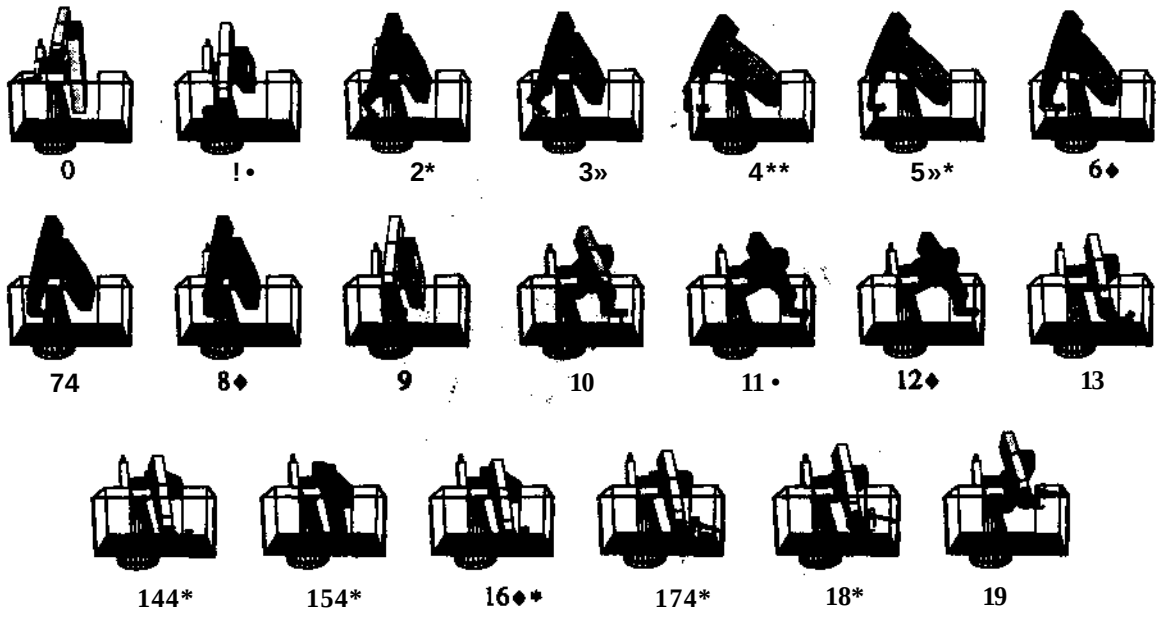
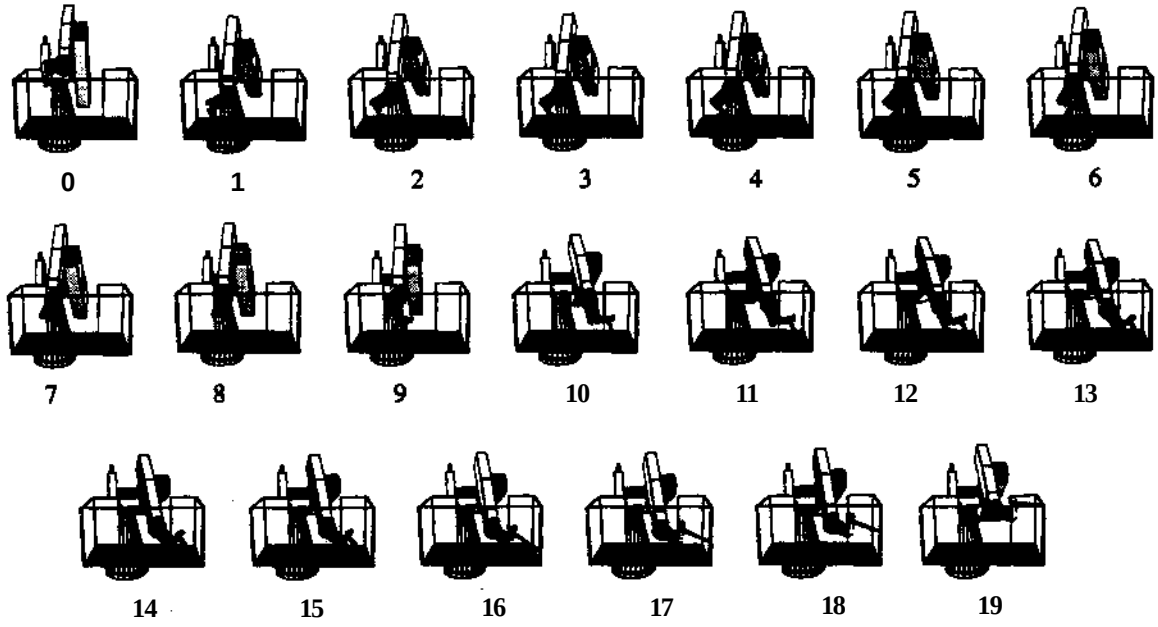



**Fig.7. Joint trajectories of PUMA762 for the case in Fig.6**




**initial solution**

**(• denotes that collision occurs at this step)**
**(* denotes that joints are outside the limits at this step)**




**final solution**

**Fig.8. Simulated results of the initial solution and the final solution in the case of Fig.6**