Kinematic Synthesis of Mechanisms and Robotic Manipulators With Binary Actuators*

G. S. Chirikjian
Department of Mechanical Engineering,
Johns Hopkins University,
Baltimore, MD 21218

Binary actuators have only two discrete states (denoted "0" and "1"), both of which are stable without feedback. As a result, binary mechanisms and manipulators have a finite number of states. Major benefits of binary actuation are that extensive feedback control is not required, task repeatability can be very high, and two-state actuators are generally very inexpensive (e.g., solenoids, pneumatic cylinders, etc.), thus resulting in low cost robotic mechanisms. This paper develops algorithms for the optimal synthesis of binary manipulators and mechanisms for discrete tasks such as pick-and-place operations.

1 Introduction

Traditional kinematic synthesis deals with the design of mechanisms to yield a desired set of motions or motion characteristics. It is generally assumed that mechanisms are actuated with continuous-range-of-motion actuators such as motors. However, there are many applications of mechanisms and robotic manipulators which only require discrete motion. For these tasks, continuous-range-of-motion machines are overkill. This indicates the need for new methods in kinematic synthesis based on discrete (binary) actuation. This paper is a first step in the development of methods which parallel those used for mechanisms and manipulators with continuous actuation (e.g., see Erdman, 1993; Mabie and Reinholtz, 1987; and references therein).

One motivation for this work is that standard robotic manipulators (which are often used for pick-and-place tasks) have not been embraced by many industries because of their relatively high cost, low accuracy, and low payload capability as compared to dedicated machine tools. Similarly, many mechanism design problems do not require continuous range-of-motion actuation, but since that is what traditional methods address, those are the only kinds of designs considered.

The synthesis methods developed in this paper represent a step in the formulation of a binary paradigm for mechanisms and robots. It is hoped that binary mechanism synthesis will lead to mechanisms and manipulators with lower cost, higher reliability, and easier connectivity to computers.

The methods presented in this paper are equally applicable to robotic manipulators and actuated mechanisms. Therefore, these terms will be used interchangeably. Likewise, the term "end-effector" will refer to any part of a mechanism or manipulator that is to be placed at desired positions.

In principle, an analogy can be made between continuous vs. binary mechanisms and analog vs. digital circuits. In the history of electronics and computing, digital devices replaced many of their analog counterparts because of higher reliability and lower cost—exactly the same reasons for developing a binary paradigm for kinematics.

A schematic of a highly actuated prototype is shown in Fig. 1 for two of its almost thirty three thousand (2^{15}) configurations. This particular design is a variable geometry truss manipulator. As currently configured, this manipulator consists of 15 identical prismatic actuators, each with two stable states (completely retracted, 0, or completely extended, 1). In this figure each cylinder has minimal and maximal lengths $q_i^{\min} = \frac{3}{20}$ and $q_i^{\max} = \frac{5}{20}$, and the width of each platform is $w_i = \frac{1}{5}$, for $i = 1, \ldots, 15$.

Actuators are numbered from left to right in each "bay" of the truss, and from base to tip. Writing these 1's and 0's from left to right, the most significant bit corresponds to the actuator on the left side of the base, and the least significant bit corresponds to the actuator at the right side of the distal end of the manipulator. Using this numbering system, the configurations shown in Fig. 1 are the "1's compliment" of each other (a term used in digital logic/Boolean algebra to denote the switching of all ones and zeros in an n-bit number so that the compliments add to $2^n - 1$).

If one reviews the literature, sporadic efforts in binary actuation can be found, e.g., (Anderson and Horn, 1967; Pieper, 1968; Roth et al. 1973). Such efforts resulted when computers were first becoming widely available to control robotic manipulators. However, despite the seemingly natural parallel between discretely actuated mechanical systems and the development of the computer, these efforts were abandoned for lack of a framework in which to design and plan well-behaved motions of such systems.

Of course, a natural question that one might raise is how different binary manipulators are from current systems which use stepper motors, or pick-and-place machines used in circuit board fabrication, or even flexible automation systems in which technicians set joint stops. The answer is that, just as in electronics, the true benefit of binary mechanisms is not so much a function of their discrete nature as it is the reliability of having only two states. Moreover, simple robots with only a few binary actuators cannot perform complicated tasks such as obstacle avoidance. Therefore, the true benefit of a binary paradigm for robotics can only be exploited if a large number of actuated degrees of freedom are considered.

The remainder of this paper is organized as follows: Section 2 formalizes and generalizes the concept of a binary manipulator. Section 3 introduces and solves the optimal design problem for binary manipulators with more kinematic parameters than desired end-effector positions. Section 4 introduces and solves

*This work was made possible by NSF Young Investigator Award IRI-9357738 from the Robotics and Machine Intelligence Program, and a Presidential Faculty Fellows Award.

Contributed by the Mechanisms Committee for publication in the JOURNAL OF MECHANICAL DESIGN. Manuscript received Feb. 1994; revised March 1995. Associate Technical Editor: G. R. Pennock.

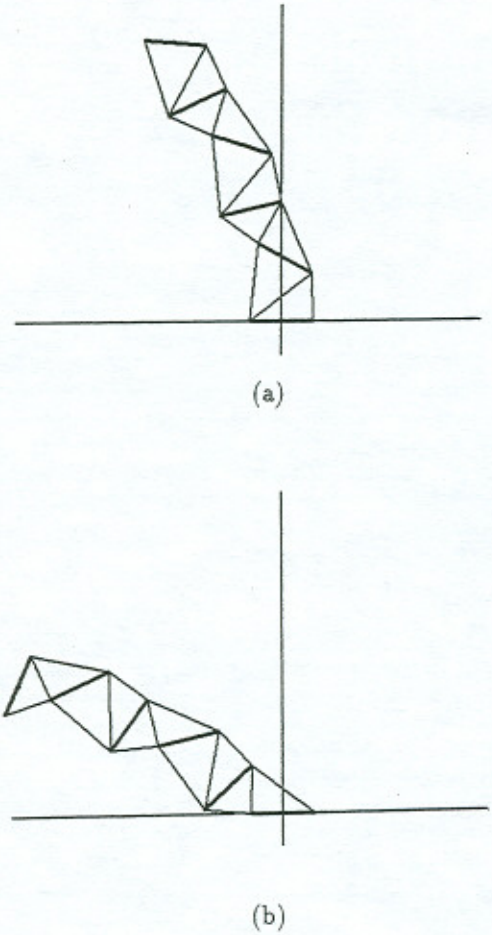

Fig. 1 Two configuration of a 15-Bit manipulator: (a) 110001110001110 (b) 001110001110001

the least squares design problem for binary manipulators with fewer kinematic parameters than desired end-effector positions. Section 5 illustrates these methods with examples.

2 Formalizing the Paradigm

The *forward kinematics* of a robotic manipulator expresses the relationship between generalized joint displacements and the position and orientation of the end-effector in space. If $\mathbf{q} = [q_1, \ldots, q_n]^T \in \mathbb{R}^n$ denotes a vector whose elements are the joint angles of a robotic manipulator, then the forward kinematic function $\mathbf{g}(\cdot) \in \mathbb{R}^N$ maps these joint angles to end-effector coordinates. This is written as:

$$\mathbf{x}_{ee} = \mathbf{g}(\mathbf{q}, \boldsymbol{\alpha}). \tag{1}$$

$\mathbf{x}_{ee} \in \mathbb{R}^N$ represents the position and/or orientation of the end-effector with respect to a given reference frame in space. For position in the plane, $N = 2$, and in space, $N = 3$. The vector $\boldsymbol{\alpha}$ contains condensed information about the kinematic structure of the manipulator. Typically, $\boldsymbol{\alpha} \in \mathbb{R}^{3n}$ for spatial serial manipulators because each link has three structural variables given by the Denavit-Hartenberg framework, i.e., link length, offset, and twist. This is usually different for parallel and hybrid parallel-serial manipulators. In essence, it is the choice of $\boldsymbol{\alpha}$ which distinguishes one member of a class of manipulators from another, while it is the part of the structure of $\mathbf{g}(\cdot)$ which is independent of $\boldsymbol{\alpha}$ which distinguishes among different classes.

For standard motor-driven robotic manipulators, each joint angle (or generalized displacement) can be controlled to achieve any desired value within a specified interval. That is,

$$q_i \in [q_i^{\min}, q_i^{\max}] \in \mathbb{R}, \tag{2}$$

where the notation $x \in [y, z]$ is equivalent to $y \leq x \leq z$. In order for a motor to achieve a desired angle, complex (and expensive) feedback control systems are usually required.

The kinematics of binary manipulators is also described by Eq. (1). The difference is that for binary manipulators,

$$q_i \in \{q_i^{\min}, q_i^{\max}\} \tag{3}$$

where $\{y, z\}$ denotes the set of two real numbers y and z. Another way to write this is for each $i \in [1, \ldots, n]$:

$$q_i = q_i^{\min} + b_{n-i}(q_i^{\max} - q_i^{\min}) \quad \text{for} \quad b_{i-1} \in \mathcal{B} \tag{4}$$

where $\mathcal{B} = \{0, 1\}$ is the set of one-bit binary numbers, or Boolean variables. A binary manipulator configuration is thus described by the n-bit binary number $b_{n-1}b_{n-2} \ldots b_1b_0 \in \mathcal{B}^n$. The most significant bit, b_{n-1}, corresponds to the first joint angle, and the least significant bit corresponds to the distal end.

Equation (4) can be written in matrix form as:

$$\mathbf{q} = \mathbf{q}^{\min} + B(\mathbf{q}^{\max} - \mathbf{q}^{\min}) \tag{5}$$

by defining the diagonal matrix B in such a way that each of its diagonal elements are a bit of the binary number with the most significant bit written in the (1, 1) place in the matrix, and least significant bit written in the (n, n) place. The vectors $\mathbf{q}^{\min}, \mathbf{q}^{\max} \in \mathbb{R}^n$ contain all joint limit information.

If the manipulator is to be used for pick-and-place tasks with multiple intermediate points defining a discrete trajectory, one possible design approach is to set the manipulator kinematic parameters $(\mathbf{q}^{\min}, \mathbf{q}^{\max}, \boldsymbol{\alpha})$ so that the end-effector can exactly reach a finite number of points in the workspace. With a priori knowledge of a finite number of points which the end-effector must traverse, algorithms presented here will generate the kinematic parameters of a binary manipulator which guarantee that the end-effector reaches the designated points. In order to solve this problem, it is first necessary to formalize the scenario.

Suppose there are m distinct points which the manipulator's end-effector is to reach corresponding to m different bit patterns. In practice, the m desired end-effector positions would be given, while the corresponding binary states would have to be found. One way to do this is with a discrete optimization procedure which yields the m binary manipulator states resulting in the end-effector positions of the baseline design closest to the desired points. In this way, the baseline kinematic parameters only need to be tweaked slightly for the manipulator to achieve the desired performance. In this paper it is assumed that the m binary states have already been chosen. For each of these configurations, we can write the expressions:

$$\mathbf{q}^i = \mathbf{q}^{\min} + B_i(\mathbf{q}^{\max} - \mathbf{q}^{\min}) \tag{6}$$

and

$$\mathbf{x}_{ee}^i = \mathbf{g}(\mathbf{q}^i, \boldsymbol{\alpha}) \tag{7}$$

for $i = 1, \ldots, m$, where B_i has been specified for each i. Equations (6) and (7) can all be written together as one big equation:

$$\begin{pmatrix} \mathbf{x}_{ee}^1 \\ \vdots \\ \mathbf{x}_{ee}^i \\ \vdots \\ \mathbf{x}_{ee}^m \end{pmatrix} = \begin{pmatrix} \mathbf{g}(\mathbf{q}^1, \boldsymbol{\alpha}) \\ \vdots \\ \mathbf{g}(\mathbf{q}^i, \boldsymbol{\alpha}) \\ \vdots \\ \mathbf{g}(\mathbf{q}^m, \boldsymbol{\alpha}) \end{pmatrix} \in \mathbb{R}^{mN}. \tag{8}$$

By defining:

$$\mathbf{a} = \begin{pmatrix} \mathbf{q}^{\min} - \mathbf{q}_0^{\min} \\ \mathbf{q}^{\max} - \mathbf{q}_0^{\max} \\ \boldsymbol{\alpha} - \boldsymbol{\alpha}_0 \end{pmatrix} \in \mathbb{R}^{Mn}, \tag{9}$$

where $M = 5$ in the serial case, Eq. (8) is written in the compact notation:

$$\mathbf{x} = \mathbf{f}(\mathbf{a}) \in \mathbb{R}^{mN}, \tag{10}$$

where $\mathbf{x}$ is called the "aggregate" end-effector position vector, and $\mathbf{a}$ is the vector of variable kinematic parameters measured from the baseline design [thus the subtraction of baseline parameters $\mathbf{q}_0^{\max}$, $\mathbf{q}_0^{\min}$, and $\boldsymbol{\alpha}_0$ in (9)].

If the only kinematic parameters which are variable are joint stops, then we write:

$$\mathbf{a} = \begin{pmatrix} \mathbf{q}^{\min} - \mathbf{q}_0^{\min} \\ \mathbf{q}^{\max} - \mathbf{q}_0^{\max} \end{pmatrix} \in \mathbb{R}^{Mn}. \tag{11}$$

where $M = 2$ in this case. Unlike the $M = 5$ case, this is independent of whether the manipulator is serial, parallel, or hybrid serial-parallel.

The form of Eq. (10) indicates that the problem of kinematic synthesis for binary manipulators performing pick-and-place tasks is analogous to the standard inverse kinematics problem for redundant continuous range-of-motion manipulators (with the roles of joint angles and kinematic parameters reversed). Furthermore, the problem is addressed differently depending on whether: $mN < Mn$, $mN = Mn$, $mN > Mn$. Adopting terminology from robot kinematics, the above cases are respectively called "redundant," "sufficient," and "insufficient." In this paper these terms are used interchangeably with: "underdetermined," "completely determined," and "overdetermined."

In the underdetermined case, it is desirable to minimize a cost function in order to resolve the redundancy. This is addressed in Section 3. In the case of equality, the expression is linearized and iterated in much the same way that kinematically sufficient manipulator joint-trajectory generation can be performed. When the problem is overconstrained, then the methods of Section 4 are used.

A binary manipulator is a semi-dedicated machine, i.e., it is designed to do a very specific pick-and-place task. However, the kinematic parameters determined by the design process can be changed as new tasks are presented to the design algorithm. In practice, these physical parameters would be set by making adjustments to the manipulator structure by changing the joint stops. This in turn specifies the geometry associated with each binary state of the manipulator. Figures 2(*a*) and (*b*) are the workspaces of the same binary manipulator shown in Fig. 1 with different joint stops. In Fig. 2(*a*), $q_i^{\min} = \frac{3}{20}$ and $q_i^{\max} = \frac{5}{20}$, whereas in Fig. 2(*b*), $q_i^{\min} = \frac{4}{25}$ and $q_i^{\max} = \frac{6}{25}$. For large numbers of bits, brute force representation of the workspace becomes intractable. An efficient alternative can be found in Ebert-Uphoff and Chirikjian (1995).

Joint Limits can be altered by putting stoppers of different lengths to limit the stroke in a desired way. This idea can be generalized to include *programmable stops* which allow the stroke to be controlled. The benefit of programmable stops is that they could operate at very low speeds. They would be fixed while a task is being performed, and adjust themselves over a much larger time scale than the duration of a task, thus adding flexibility of function and form without an explosive increase in cost.

3 Optimal Binary Mechanism Synthesis: Underdetermined Case

As was discussed in Section 2, we are presented with a problem in which the number of variable kinematic parameters exceeds the number of end-effector position variables that we seek to specify. This is analogous to the redundancy resolution problem in manipulator kinematics.

Fig. 2 Workspace of 15-bit manipulator with: (*a*) $(q_i^{\min}, q_i^{\max}) = (3/20, 5/20)$ (*b*) $(q_{\min}, q_{\max}) = (4/25, 6/25)$; for $i = 1, \ldots, 15$

In this section, methods for kinematic redundancy resolution are reformulated to derive equations governing the optimal adjustment of binary manipulator kinematic parameters. The criteria used here are intuitive: given an initial, or baseline, design and a number of points which the manipulator is expected to reach, alter the original design as little as possible in order to reach the points. Thus, the vector $\mathbf{x}$, which is the concatenation of end-effector positions corresponding to each of the m initial configurations, will be made to follow an imaginary trajectory to the desired aggregate end-effector positions. This trajectory will be parametrized by "artificial time" (i.e., we are not interested in the trajectory that links the baseline and final aggregate end-effector positions, but rather only the end points). Since this calculation is done off line and not in real time, problems with computational efficiency do not drive the development of algorithms in the binary synthesis problem to the extent they do in redundancy resolution. A problem which is similar to redundancy resolution is that the variation of the kinematic parameters (joint stops) in artificial time must not only follow end-effector constraints, but also satisfy optimality criteria which are path independent (since we are only interested in terminal values). This is called the *cyclicity* problem in redundancy resolution, and has been addressed in a number of works, e.g. (Baker and Wampler, 1988; Chung et al., 1994).

This section is organized as follows: Section 3.1 reviews the derivation of the Jacobian pseudo-inverse solution and shows how this is applied to the kinematic synthesis of binary mechanisms. Section 3.2 reformulates this method in order to synthe-

size binary mechanisms in a way that makes the smallest possible changes to a baseline design.

3.1 Incremental Optimal Synthesis. The most common redundancy resolution technique is adapted here to find incremental changes in the vector $\mathbf{a}$ so as to alter a baseline design in a controlled way while observing the desired aggregate end-effector position vector. Incremental changes in $\mathbf{a}$ are denoted $\delta\mathbf{a}$, and incremental changes in aggregate end-effector position are denoted as $\delta\mathbf{x}$.

In terms of incremental changes, the aggregate end-effector constraint (10) is written as:

$$\delta\mathbf{x} = J\delta\mathbf{a}, \tag{12}$$

where $J = \mathbb{R}^{mN\times Mn}$ is the Jacobian of the transformation. It is assumed throughout this section that $mN < Mn$, i.e., that the problem is underdetermined.

Let us assume that it is desirable to find $\delta\mathbf{a}$, which will minimize the quantity:

$$\tfrac{1}{2}\delta\mathbf{a}^T W\,\delta\mathbf{a} \tag{13}$$

from point to point, subject to the constraint that the aggregate end-effector vector follows a prescribed trajectory from initial to final points. W is symmetric and positive definite. The rationale for this formulation is twofold: (1) efficient existing techniques are easily adapted; (2) in practice incremental minimization of $\delta\mathbf{a}$ yields reasonable values for $\mathbf{a}$ when the path between baseline and goal aggregate end-effector locations is direct, and the baseline design does not need to be altered substantially.

This problem is solved using standard methods of constrained optimization (Campbell and Meyer, 1979), which yield

$$\delta\mathbf{a} = W^{-1}J^T(JW^{-1}J^T)^{-1}\delta\mathbf{x}. \tag{14}$$

Note that when $W = 1$, the standard Moore-Penrose pseudoinverse results.

Thus, the vector $\mathbf{x}$, which is the concatenation of end-effector positions corresponding to m different binary configurations, will be made to follow a trajectory to the desired aggregate end-effector positions. This trajectory will be parameterized by "artificial time" (i.e., we are not interested in following a trajectory, but rather the resulting end points). This calculation is done off line and not in real time. Since this method is *not* cyclic, and the results will therefore be path dependent, the trajectory is chosen heuristically to be a straight line connecting baseline and goal aggregate end-effector positions. This is written as:

$$\mathbf{x}(t) = \mathbf{x}_b + t(\mathbf{x}_g - \mathbf{x}_b), \tag{15}$$

where $\mathbf{x}_b$ and $\mathbf{x}_g$ are the baseline and goal aggregate end-effector positions, respectively. Alternate approaches based on other redundancy resolution techniques can also be used (e.g., Baillieul, 1985; Nakamura et al., 1987). Ideally, the variation of the kinematic parameters (joint stops) in artificial time must not only follow end-effector constraints, but also satisfy optimality criteria which are *path independent* since we are only interested in terminal values. The pseudoinverse approach can be augmented so the resulting solution is path independent, as discussed in the literature (e.g., Baker and Wampler, 1988; Chung et al., 1994). The following subsection presents one approach.

3.2 Configuration-Based Optimization: Underdetermined Case. Suppose instead of the usual joint rate norm cost functions used in kinematic redundancy resolution, we seek to minimize:

$$\tfrac{1}{2}\mathbf{a}^T W\mathbf{a} \tag{16}$$

subject to the constraints:

$$\mathbf{x} = \mathbf{f}(\mathbf{a}), \tag{17}$$

where $\mathbf{f}(\cdot)$ is the concatenation of forward kinematic functions corresponding to different binary states of the manipulator, and $W \in \mathbb{R}^{Mn\times Mn}$ is a constant symmetric positive definite matrix.

To solve the problem, a function is defined as:

$$F = \tfrac{1}{2}\mathbf{a}^T W\mathbf{a} + \boldsymbol{\mu}^T(\mathbf{f}(\mathbf{a}) - \mathbf{x}), \tag{18}$$

where $\boldsymbol{\mu}$ is a vector of Lagrange multipliers.

The constrained optimization problem then becomes one of solving the following simultaneous equations:

$$\frac{\partial F}{\partial \mathbf{a}} = \mathbf{0}^T \quad \frac{\partial F}{\partial \boldsymbol{\mu}} = \mathbf{0}^T, \tag{19}$$

which together represent a system of $Mn + mN$ scalar equations. These equations are written as Eq. (17) together with:

$$W\mathbf{a} + J^T\boldsymbol{\mu} = \mathbf{0}, \tag{20}$$

where $J \in \mathbb{R}^{mN\times Mn}$ is again the Jacobian matrix $\partial\mathbf{f}/\partial\mathbf{a}$.

Equations (17) and (20) can be written together in rate, or incremental, form as:

$$\begin{pmatrix} \hat{W} & J^T \\ J & O \end{pmatrix}\begin{pmatrix} \delta\mathbf{a} \\ \delta\boldsymbol{\mu} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \delta\mathbf{x} \end{pmatrix}. \tag{21}$$

$\hat{W} = \hat{W}^T$ is a matrix such that

$$\hat{W}\delta\mathbf{a} = W\delta\mathbf{a} + \delta J^T\boldsymbol{\mu}. \tag{22}$$

By inverting the above matrix, we can solve the following simultaneously:

$$\delta\mathbf{a} = \hat{W}^{-1}J^T(J\hat{W}^{-1}J^T)^{-1}\delta\mathbf{x} \tag{23}$$

and

$$\delta\boldsymbol{\mu} = -(J\hat{W}^{-1}J^T)^{-1}\delta\mathbf{x}. \tag{24}$$

The kinematic parameters and Lagrange multipliers are updated with the simple rule $\mathbf{a} \leftarrow \mathbf{a} + \delta\mathbf{a}$ and $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \delta\boldsymbol{\mu}$. The initial values of these vectors are both zero because at the baseline design $\mathbf{a} = \mathbf{0}$ and the constraints need not be enforced, so $\boldsymbol{\mu} = \mathbf{0}$.

Due to numerical inaccuracy, this method can be augmented so that the right hand side of (21) is replaced with:

$$\begin{pmatrix} -\alpha\left(\dfrac{\partial F}{\partial \mathbf{a}}\right)^T \\ \delta\mathbf{x} \end{pmatrix}. \tag{25}$$

$(\partial F/\partial\mathbf{a})^T$ is the left side of (20). In general, $0 < \alpha < 1$ is a scalar "gain." Thus, if the solution is accurate, this term will vanish because (20) holds, but if numerical errors are present, this term will provide corrective feedback to converge to the solution. This result is known in the optimization literature as the Lagrange-Newton method.

4 Kinematically Insufficient Binary Mechanisms

In the previous section it was assumed that the number of adjustable kinematic parameters was equal to, or exceeded, the number of end-effector position variables which the binary manipulator was expected to accommodate exactly. However, if we want to use a greater fraction of the end-effector positions attainable with binary manipulators, then the following kinematically "insufficient" problem must be considered: Given that the number of specified end-effector position variables (mN) exceeds the number of adjustable kinematic parameters (Mn), alter the binary manipulator geometry so that the end-effector

positions come *as close as possible* to the desired points. Since the number of end-effector points generated with an n-bit binary number is 2^n, whereas the number of kinematic parameters is linear in the number of joints (i.e., Mn), the kinematically insufficient problem is inescapable as n increases.

This section formulates this issue using methods from the theory of generalized inverses (e.g., Golub and Van Loan, 1983; Campbell and Mayer, 1979). The basic approach is the same as in Chirikjian (1994b). Sections 4.1 and 4.2 parallel the solutions presented in the previous section for the overdetermined case. In Subsection 4.1, the method of least squares is used in an iterative fashion, in an attempt to distribute the inaccuracy inherent in solutions to overconstrained problems evenly over the end-effector points. In Subsection 4.2, a scheme based on configuration optimization for overconstrained systems is developed.

4.1 An Iterative Least-Squares Approach. In this section, the method of least squares is used to alter a binary manipulator's kinematic parameters so that the end-effector positions corresponding to chosen binary states come as close as possible to what is desired. Because the number of kinematic parameters available to vary is less than the number of end-effector position variables which the manipulator is expected to reach, there will generally be no exact solution. The best we can hope for in this kinematically insufficient scenario is to minimize error.

Recall the general overdetermined least squares problem: Given a matrix $A \in \mathbb{R}^{r \times s}$ with rank s, where $r > s$, and vectors $\mathbf{y} \in \mathbb{R}^s$, $\mathbf{b} \in \mathbb{R}^r$, then the weighted least squares solution of the problem

$$A\mathbf{y} = \mathbf{b} \tag{26}$$

is the one for which the weighted 2-norm of: $\mathbf{e} = A\mathbf{y} - \mathbf{b}$ (the error vector) is minimized. Using elementary techniques of unconstrained minimization, i.e., set the gradient of a cost function to zero ($\partial(\mathbf{e}^T W \mathbf{e})/\partial \mathbf{y} = \mathbf{0}^T$) it is easy to show that the solution to this problem has the form:

$$\mathbf{y} = (A^T W A)^{-1} A^T W \mathbf{b} = A_W^+ \mathbf{b}, \tag{27}$$

where A_W^+ denotes the weighted least squared error pseudoinverse.

Recall from the previous sections that the kinematic synthesis problem for binary mechanisms lends itself to the form:

$$J\delta\mathbf{a} = \delta\mathbf{x}. \tag{28}$$

In the current scenario, the number of rows of J exceeds the number of columns because there are fewer kinematic parameters than aggregate end-effector constraints.

Equation (27) represents an expression which must be iterated with $A = J$, $\mathbf{y} = \delta\mathbf{a}$, and $\mathbf{b} = \delta\mathbf{x}$ to yield a solution to the problem at hand. Starting with initial values: $\mathbf{a}(0) = \mathbf{0}$ and $\mathbf{x}(0) = \mathbf{x}_b$, the kinematic parameters are augmented at each timestep in artificial time, t, as:

$$\mathbf{a}(t + \delta t) = \mathbf{a}(t) + \delta\mathbf{a}(t), \quad \delta\mathbf{a} = J_W^+ \delta\mathbf{x}. \tag{29}$$

Even though $\delta\mathbf{a}$ as calculated above is the "best possible" solution, each iteration will result in the end-effector drifting away from the prescribed "imaginary" trajectory, $\mathbf{x}(t)$, because the problem is overconstrained. This is remedied in a straightforward way: instead of defining $\delta\mathbf{x}$ to be the tangent associated with a desired imaginary trajectory ($\delta\mathbf{x} = (d\mathbf{x}/dt)\delta t = (\mathbf{x}_g - \mathbf{x}_b)\delta t$), use the heuristic definition:

$$\delta\mathbf{x} = F(\|\mathbf{x}_g - \mathbf{f}(\mathbf{a})\|)(\mathbf{x}_g - \mathbf{f}(\mathbf{a}))\delta t \tag{30}$$

where again $\mathbf{x}_g$ is the aggregate goal end-effector position, and $F(\cdot)$ is a function that is chosen to accelerate the convergence to the goal. Two possible choices are $F(x) = \beta$ and $F(x) = \beta/\sqrt{\epsilon^2 + x^2}$, where ϵ is a small constant introduced to prevent division by zero if the goal is achieved, and β is a constant "gain" used to regulate convergence to the goal.

In this way, the solution is continually forced in the direction of the goal, even though no a priori trajectory is followed. For the problem at hand, unlike the kinematic redundancy resolution problem, reaching the goal is the only task of significance, and the "imaginary" trajectory need not be followed.

Another problem to overcome is the potential loss of rank in J—the aggregate jacobian, and the resulting numerical problems in computing the pseudo-inverse. To avoid this problem, and to bound the magnitude of $\delta\mathbf{a}$, a "damped" version of the pseudo-inverse is used:

$$\delta\mathbf{a} = (J^T W J + \rho^2 1)^{-1} J^T W \delta\mathbf{x}, \tag{31}$$

where ρ is an introduced constant. This solution results from the minimization of the cost function:

$$C = \mathbf{e}^T W \mathbf{e} + \rho^2 \delta\mathbf{a}^T \delta\mathbf{a} \tag{32}$$

which bounds the norm of $\delta\mathbf{a}$, where $\mathbf{e} = J\delta\mathbf{a} - \delta\mathbf{x}$.

4.2 Configuration-Based Optimization: Overdetermined Case. In the previous subsection, we sought to minimize the instantaneous (or rate) error: $\|J\delta\mathbf{a} - \delta\mathbf{x}\|$ at each iteration. The benefits of that approach are conceptual simplicity, easy implementation with standard techniques, and relatively little computational complexity. The drawbacks of that approach are that it does not minimize $\mathbf{a}^T\mathbf{a}$ (even though it does bound $\delta\mathbf{a}^T\delta\mathbf{a}$ in the damped case), and the solution may not converge. The former of the above problems can lead to designs that vary greatly from the baseline design, and the latter indicates that no solution may result.

In this subsection, the direct problem of minimizing $\|\mathbf{f}(\mathbf{a}) - \mathbf{x}\|$ is addressed for the overdetermined case. Thus, the developments of this and the previous subsection parallel the rate and configuration based methods presented in the previous section.

In order to generate the desired results, a cost function of the form:

$$C = \tfrac{1}{2}(\mathbf{f}(\mathbf{a}) - \mathbf{x})^T M(\mathbf{f}(\mathbf{a}) - \mathbf{x}) + \tfrac{1}{2}\mathbf{a}^T W \mathbf{a} \tag{33}$$

is minimized, where W and M are constant symmetric positive definite matrices. These matrices will generally be taken to be diagonal. The purpose of introducing the matrix W is to weight the sensitivity of kinematic parameters with respect to each other. The matrix M weights the sensitivity of end-effector position error. Ratios of the norms of the two matrices allows us to balance the total amount of change to the baseline design and aggregate end-effector error.

Setting the gradient of C defined in Eq. (33) to zero, one gets:

$$\left(\frac{\partial \mathbf{f}}{\partial \mathbf{a}}\right)^T M(\mathbf{f}(\mathbf{a}) - \mathbf{x}) + W\mathbf{a} = \mathbf{0}. \tag{34}$$

This is a nonlinear algebraic equation which is always difficult (and usually impossible) to solve analytically. The above can be linearized by considering infinitesimal changes:

$$\left[\frac{\partial^2 \mathbf{f}^T}{\partial \mathbf{a}^2} M(\mathbf{f} - \mathbf{x}) + \left(\frac{\partial \mathbf{f}}{\partial \mathbf{a}}\right)^T M \frac{\partial \mathbf{f}}{\partial \mathbf{a}} + W\right]\delta\mathbf{a}$$

$$= K\delta\mathbf{a} = \left(\frac{\partial \mathbf{f}}{\partial \mathbf{a}}\right)^T \delta\mathbf{x}, \tag{35}$$

and inverting K to find $\delta\mathbf{a}$ that will yield the desired solution.

The elements of the matrix K are written explicitly below so that there is no confusion about the above notation:

$$K_{ij} = W_{ij} + \sum_{k,l=1}^{mN} \left(\frac{\partial^2 f_k}{\partial a_i \partial a_j} M_{kl}(f_l - x_l) + \frac{\partial f_k}{\partial a_i} M_{kl} \frac{\partial f_l}{\partial a_j} \right). \quad (36)$$

Positive definiteness of the matrix K is the sufficient condition for a minimal solution. If at the beginning of the process we check for positive definiteness, and the matrix remains invertible during the process (i.e., none of the eigenvalues become nonpositive) then the matrix is guaranteed to remain positive definite. Thus, the algorithm is guaranteed to provide an optimal solution provided the matrix above is initially positive definite, and remains invertible throughout the process.

If a singularity is encountered, the linear trajectory defined initially must be heuristically augmented to attempt to circumvent the singularities. If this is not possible, the method will not converge. However, given that the trajectory lengths are extremely small in comparison to manipulator length scales, one would expect that algorithmic singularities will be rare.

5 Examples of Binary Mechanism Synthesis

In this section, the synthesis algorithms presented in Sections 3 and 4 are demonstrated with a variable geometry truss structure. The forward kinematics of this structure can be found in Chirikjian (1994a), while the forward kinematics of other truss structures is addressed in Jain and Kramer (1990).

Subsection 5.1 demonstrates two algorithms (incremental optimization and configuration optimization) for the case when the number of desired end-effector position variables is less than the number of variable kinematic parameters. Subsection 5.2 demonstrates two algorithms (iterative incremental least squares, and configuration least squares optimization) applied to the overdetermined case.

5.1 Underdetermined Problems. Let us suppose that our goal is to move a single point of the initial workspace to a desired point in a new workspace of a 15 bit binary VGT without regard to any other points. Thus, $mN = 1 \cdot 2 = 2$ for this problem. Likewise, $Mn = 2 \cdot 15 = 30$. However, since each actuator is either fully retracted or fully extended (with no possibility of both happening at the same time), there are effectively only 15 kinematic parameters that we can change once the binary state is specified.

Applying the incremental algorithm from Section 3 (with $W = 1$) starting with the configuration in Fig. 1(a) as the baseline, and altering kinematic parameters so that the end-effector reaches the point $\mathbf{x}_{ee} = (0, 0.8)$ while retaining the same binary values, the resulting configuration is that shown in Fig. 3(a). The new joint stops are:

$$\mathbf{q}^{\min} = \begin{pmatrix} 0.150, 0.150, 0.125, 0.152, 0.155, 0.150, 0.150, 0.150, \\ 0.135, 0.145, 0.147, 0.150, 0.150, 0.150, 0.139 \end{pmatrix}^T$$

and

$$\mathbf{q}^{\max} = \begin{pmatrix} 0.256, 0.262, 0.250, 0.250, 0.250, 0.231, 0.246, 0.258, \\ 0.250, 0.250, 0.250, 0.241, 0.244, 0.255, 0.250 \end{pmatrix}^T.$$

Note that for each element of the above vectors, if the ith element has been changed in one, it is not changed in the other.

Applying the configuration optimization algorithm from Section 3 (with $W = 1$) again starting with the configuration in Fig. 1 as the baseline, and altering kinematic parameters so that the end-effector reaches the point $\mathbf{x}_{ee} = (0, 0.8)$ while retaining the same binary values. The new joint stops are:

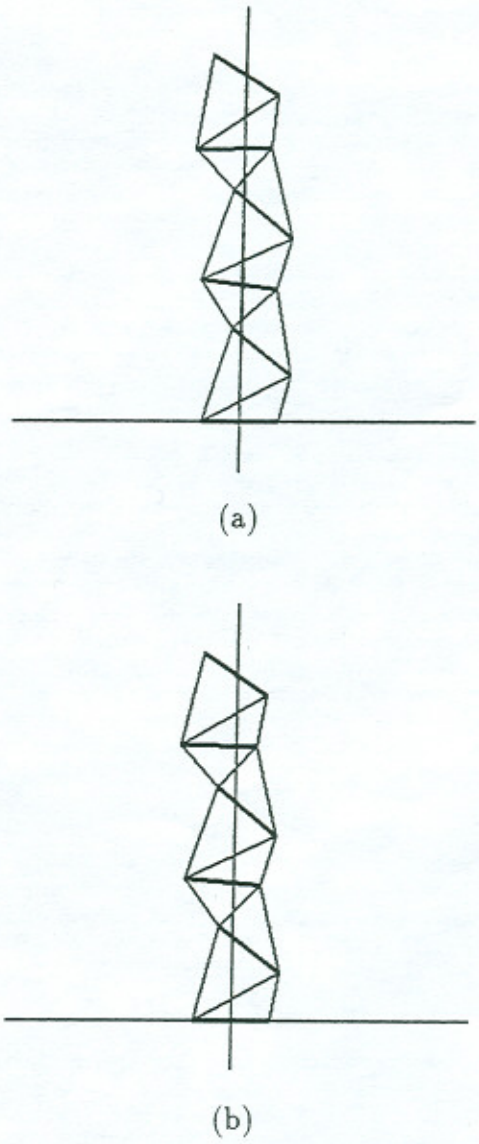

Fig. 3 Configuration generated with: (a) incremental optimization (b) configuration-based optimization

$$\mathbf{q}^{\min} = \begin{pmatrix} 0.150, 0.150, 0.129, 0.154, 0.151, 0.150, 0.150, 0.150, \\ 0.132, 0.148, 0.149, 0.150, 0.150, 0.150, 0.137 \end{pmatrix}^T$$

and

$$\mathbf{q}^{\max} = \begin{pmatrix} 0.253, 0.258, 0.250, 0.250, 0.250, 0.231, 0.248, 0.260, \\ 0.250, 0.250, 0.250, 0.238, 0.244, 0.258, 0.250 \end{pmatrix}^T,$$

which are very similar to the results generated using the incremental method. The corresponding configuration is shown in Fig. 3(b), which looks almost identical. In fact, if one is looking for a "reasonable" solution the incremental approach will come close to the configuration optimization approach when the magnitude of the difference between baseline and desired end-effector positions is small.

5.2 Overdetermined Problems. Let us suppose that we have a 3 bit binary VGT, i.e., only one section of the VGT from Fig. 1, and the transverse elements have fixed length. Thus, $Mn = 2 \cdot 3 = 6$. Furthermore, let us assume that we are interested in positioning the four end-effector points corresponding to the binary values (010, 000, 110, 111).

Figure 4 illustrates the least squares approaches presented in Section 4 for this problem. In Fig. 4(a) the baseline set of configurations has joint stops $q_i^{\min} = 0.75$ and $q_i^{\max} = 1.25$ for $i = 1, 2, 3$. The configurations shown correspond to the binary values: (010), (000), (110), and (111). The desired locations for the "end-effector" (middle of upper transverse element in the platform) corresponding to these binary values are denoted with spots. The coordinates of these locations are: (0.0, 0.8), (−0.5, 0.5), (0.1, 1.05), and (−0.4, 1.05). Using the iterative least squares approach (damped with $\rho = 0.01$ and unweighted), the resulting configurations are shown in Fig. 4(b), and the new joint stops are: $(q_1^{\min}, q_1^{\max}) = (0.934, 1.283)$, $(q_2^{\min}, q_2^{\max}) = (0.350, 1.190)$, $(q_3^{\min}, q_3^{\max}) = (0.683, 1.104)$. As can be seen, there is an improvement in the position of the end-effector, but the positions are not exact and the error is not distributed evenly.

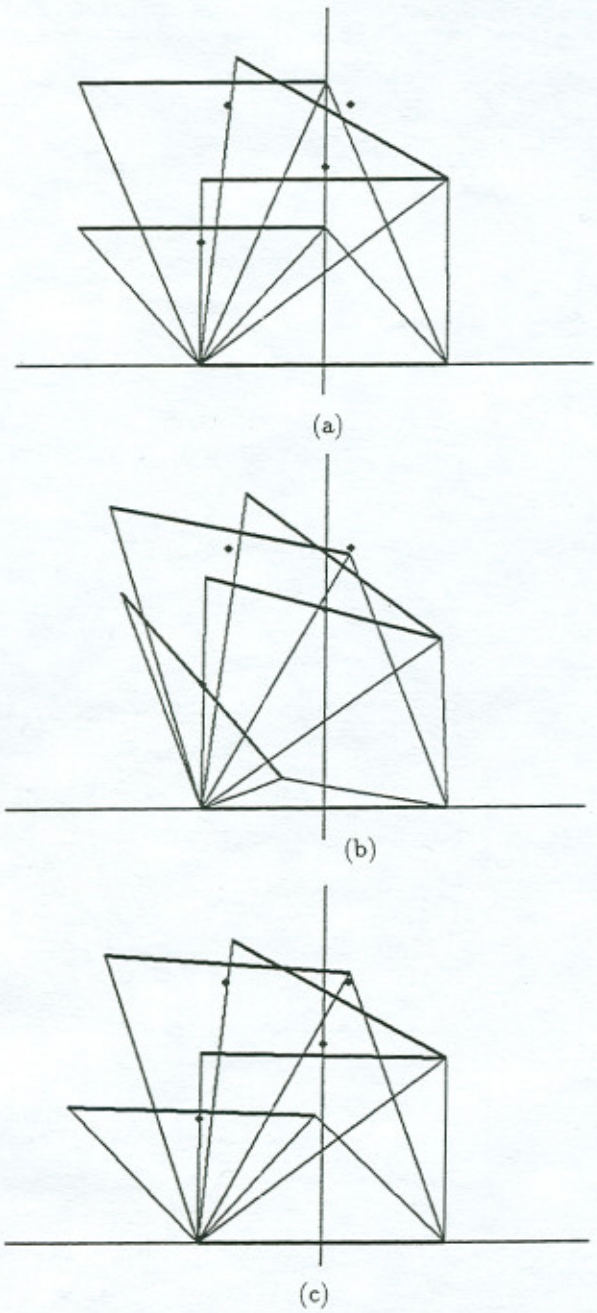

Fig. 4 4 States of a 3-Bit truss: (a) baseline; (b) altered with iterative least squares; (c) altered with configuration-based optimization

In Fig. 4(c), the same baseline kinematic parameters are used. Only now, the configuration-based error minimization method is used, where $\rho = 0.01$ and both weighting matrices (W and M) are identity matrices. The resulting joint stops are: $(q_1^{\min}, q_1^{\max}) = (0.765, 1.230)$, $(q_3^{\min}, q_3^{\max}) = (0.696, 1.245)$, $(q_2^{\min}, q_2^{\max}) = (0.743, 1.158)$. As can be seen from Fig. 4(c), the error is distributed very evenly over the end-effector points as compared to Fig. 4(b).

Figure 5(a) has the same baseline parameters as Fig. 4(a). However, it is assumed that only three points need to be reached by the end-effector. This makes the aggregate jacobian square, and so direct inversion is generally possible. Note that all the methods presented here will yield similar results (assuming ρ is small) in the sufficient case. In this example, the chosen binary values are: (010), (000), and (111). The desired locations for the end-effector corresponding to these binary values are: (0.0, 0.8), (−0.5, 0.5), and (−0.4, 1.05). The resulting configurations reach the desired points to three decimal places with joint stops: $(q_1^{\min}, q_1^{\max}) = (0.930, 1.144)$, $(q_2^{\min}, q_2^{\max}) = (0.369, 1.190)$, $(q_3^{\min}, q_3^{\max}) = (0.671, 1.104)$, and the resulting binary platform mechanism is shown in Fig. 5(b).

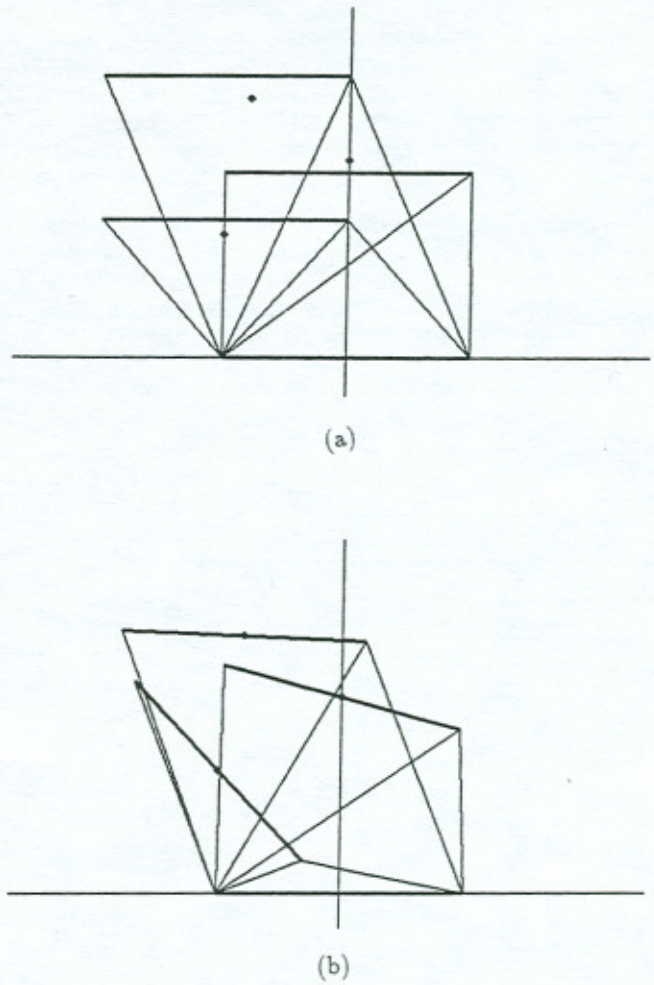

Fig. 5 3 States of a 3-bit truss: (a) baseline (b) altered configuration (exactly determined) case

6 Conclusions

This paper has presented the binary mechanism synthesis problem and provided several ways to solve it. All these methods are based on looking at the kinematic parameters of the mechanism/manipulator as the variables in the problem, and using methods of constrained optimization/redundancy resolution to yield a solution. Examples were given to illustrate these methods.

The problem solved in this paper is analogous to the shape synthesis problem in classical kinematics. It should be noted that there is an additional "layer" to the binary synthesis problem which must be addressed in future work. Namely, when the number of binary states becomes large, selecting an optimal subset of the 2^n possible configurations from which to construct the aggregate end-effector vector is a nontrivial problem for large n.

7 References

Anderson, V. C., and Horn, R. C., 1967, "Tensor Arm Manipulator Design," ASME paper 67-DE-57.

Baillieul, J., 1985, "Kinematic Programming Alternatives for Redundant Manipulators," *Proc. IEEE Int. Conf. on Robotics and Automation*, St. Louis, MO, March 1985, pp. 722–725.

Baker, D. R., and Wampler, C. W., 1988, "On the Inverse Kinematics of Redundant Manipulators," *International Journal of Robotics Research*, Vol. 7, No. 2, pp. 3–21.

Campbell, S. L., Meyer, C. D. Jr., 1979, *Generalized Inverses of Linear Transforms*, Pitman, London.

Chirikjian, G. S., 1994a, "A Binary Paradigm for Robotic Manipulators," *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, San Diego, CA.

Chirikjian, G. S., 1994b, "Kinematic Synthesis of Mechanisms and Robotic Manipulators with Binary Actuators," *Proc. 1994 ASME Mechanisms Conference*, Minneapolis, Min.

Chung, Y. S., Griffis, M. W., Duffy, J., 1994, "Repeatable Joint Displacement Generation for Redundant Robotic Systems," ASME JOURNAL OF MECHANICAL DESIGN, Vol. 116, March, pp. 11–16.

Ebert-Uphoff, I., and Chirikjian, G. S., 1995, "Efficient Workspace Generation for Binary Manipulators with Many Actuators," *Journal of Robotic Systems*, June.

Erdman, A. G., ed., 1993, *Modern Kinematics: Developments in the Last Forty Years*, Wiley, New York.

Golub, G. H., and Van Loan, C. F., 1983, "*Matrix Computations*, The Johns Hopkins University Press, Baltimore.

Jain, S., and Kramer, S. N., 1990, "Forward and Inverse Kinematic Solution of the Variable Geometry Truss Robot Based on an N-Celled Tetrahedron-Tetrahedron Truss," ASME JOURNAL OF MECHANICAL DESIGN, Vol. 112, March.

Mabie, H. M., and Reinholtz, C. F., 1987, *Mechanisms and Dynamics of Machinery, 4th ed.*, Wiley, New York.

Nakamura, Y., Hanafusa, H., and Yoshikawa, T., 1987, "Task-Priority Based Redundancy Control of Robot Manipulators," *International Journal of Robotics Research*, Vol. 6, No. 2, pp. 3–15.

Pieper, D. L., 1968, "The Kinematics of Manipulators under Computer Control," PhD Dissertation, Stanford University, Oct.

Roth, B., Rastegar, J., and Scheinman, V., 1973, "On the Design of Computer Controlled Manipulators," First CISM-IFTMM Symp. on Theory and Practice of Robots and Manipulators, pp. 93–113.