Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2006), *20*, 313–328. Printed in the USA.

DOI: 10.1017/S0890060406060239

The representation and handling of constraints for the design, analysis, and optimization of high speed machinery

B.J. HICKS, A.J. MEDLAND, AND G. MULLINEUX

Innovative Manufacturing Research Centre, Department of Mechanical Engineering, University of Bath, Bath, United Kingdom

(RECEIVED November 2, 2005; ACCEPTED July 6, 2006)

Abstract

High speed machinery has played and continues to play a vital role in the manufacture and production of consumer goods. In the design of high speed systems there are two key considerations: power transmission and motion control. Although there is considerable computer-based support for the design of systems to achieve requirements of power transmission, there is only limited support for the design of systems to deliver complex motion control. This is particularly the case where mechanism and linkage systems are considered in order to achieve large displacements and intricate paths involving reentrant and reciprocating components. One explanation for this relative lack of supportive tools is the underlying reasoning and analysis techniques implemented within many commercial and research software environments. To overcome these limitations a constraint-based approach has been employed to provide the fundamental elements of a design environment for mechanisms and machine systems. The design environment provides support for the transition from concept to embodiment stages of the design process and the subsequent stages of detailed design and optimization. In contrast to many research approaches the design environment presented in this paper has been created and developed through close collaboration with industry and through extensive application to real design scenarios. First, the underlying representations and methods are presented. The fundamental elements of the design environment are then described and its capabilities discussed with particular reference to the use of constraints in design.

Keywords: Constraints; Design; Mechanisms; Optimization; Simulation

1. INTRODUCTION

The importance of machine systems for the production and manufacture of consumer goods is widely acknowledged (MTA, 2005; US Census Bureau, 2002). The evolution of machinery, and in particular high speed machinery, can be traced from the early designs of Leonardo da Vinci and his contemporaries (Cianchi, 1988) to the large-scale high speed production lines, common in today's fast-moving consumer goods sectors. Although scale and complexity may have changed, the fundamental principles of the individual assemblies and subassemblies remain largely unaltered (Hicks et al., 2002*a*).

Reprint requests to: G. Mullineux, Innovative Manufacturing Research Centre, Department of Mechanical Engineering, University of Bath, Bath BA2 7AY, UK. E-mail: g.mullineux@bath.ac.uk

Because of the dependence of the manufacturing industry on high speed machinery, the subject of machine design has received considerable attention over the years, from both academia (Timoshenko & Young, 1948; Shigley & Uicker, 2003) and the vendors of commercial software systems (MSC Software Corporation, 2005; PTC, 2005; UGS Corporation, 2005). The main focus of much of today's academic research effort has been on the creation of improved design processes and analysis techniques for particular classes of machine. This includes, for example, rotating and linear power transmission systems (Counsell et al., 1999; Hicks et al., 2005*a*), mechanisms (Camlinks Ltd., 2005), and hydraulic circuits (Richards et al., 1999). In general, these techniques have been created to improve the specification and selection of particular components to meet the desired power transmission requirements. In addition, the computational techniques may also help visualize

the designed system (Visual Components Ltd., 2005), which is particularly useful where there are a large number of subassemblies that interact concurrently with each other and also the product (Hicks et al., 2002*b*).

In contrast to the wealth of support for designing systems based on loading requirements, there are relatively few techniques that support synthesis and embodiment to achieve desired motion characteristics. This is particularly the case where complex motion paths are required and complicated mechanisms are considered. It should be noted that in the domains of hydraulics and mechatronics, motion control can also be specified in conjunction with loading. However, this tends to be limited to straightforward linear and purely rotational motion.

For the purpose of this paper, complex motion paths are considered to include motion paths that involve large displacement, asymmetric revolution, reentrant motion, and reciprocating elements. This may include cams, linkages, gears, and mechanical couplings. Figure 1 shows a number of examples of high speed machines that incorporate a range of complex motion paths to achieve their function. Where these complex motion paths are considered, the designer is concerned with determining suitable sizes, combinations, and relative position of components necessary to achieve the desired output motion. The current design process employed within many organizations involves sketching out preliminary designs and then, either manually or within a computer-aided design (CAD) system, plotting the system configuration at various points throughout its operational cycle in order to evaluate the motion. This evaluation may also consider velocities, accelerations, and jerk (the third derivative with respect to time). This process can be both time consuming and analytically intensive, particularly when iteration is necessary to refine and improve design solutions. Furthermore, with today's large-scale complex systems, the designer also needs to consider the multitude of interactions between all the subsystems. This again demands that the designer propagate and evaluate changes to one subsystem with respect to all the other subsystems.

For the reasons previously outlined, computational support for the design, analysis, and optimization of high speed machinery is particularly desirable. To address this issue a number of attempts have been undertaken (Camlinks Ltd., 2005; PTC, 2005; UGS Corporation, 2005). However, these systems are generally limited in their capability to support design synthesis and subsequent system optimization. Furthermore, the environments may be limited in terms of the number of elements, assemblies, and component types that may be considered. The fundamental factor that limits these environments can be considered to be the underlying for-

Fig. 1. Examples of high speed machine systems involving complex motion. [A color version of this figure can be viewed online at www.journals.cambridge.org]

https://doi.org/10.1017/S0890060406060239 Published online by Cambridge University Press

mulation used to simulate and analyze the system model. For example, simulation code may only exist for predetermined configurations, or sequential parametric representations may be employed. The latter of these means that a predefined order of system resolution is imposed that frustrates the ability of the designer to explore the design envelope, elicit important design constraints, investigate design alternatives, and undertake design synthesis.

To overcome some of the limitations imposed by underlying computational or numerical methods, constraint-based techniques have been employed. For example, constraints have been successfully used in the areas of conceptual design (O'Sullivan, 2002*a*), configuration design (Mailharro, 1998; Stumptner et al., 1998), process planning (Markus et al., 2002; Armarego & Ostafiev, 2003), assembly modeling (Anantha et al., 1996), and linking function and grammar (Schmidt et al., 2005). Such application areas are characterized by the requirement to handle evolving design principles (structure), changing design goals (including global and local objective functions), altering constraint sets (networks), and the need to computationally determine a solution that satisfies the constraint set or at least provide a measure of the ability of a solution to satisfy the constraint set. All of these elements are also important for synthesizing and optimizing high speed machinery. For these reasons constraint-based modeling techniques have been adapted to provide the fundamental reasoning and analysis elements of a design environment for high speed machinery. Central to the development of this design environment has been its extensive use by a variety of engineering organizations and consultancies (Gray & Atalan, 2002; Hicks et al., 2002*a*; Fine Systems Corporation Limited, 2005). The fundamental elements of the environment are presented in detail in this paper. In particular, the underlying representations and methods are summarized, and the functionality and capabilities of the design environment are discussed with particular reference to the use of constraints for design. The first sections summarize the underlying methods and the core elements of the design environment. Following this, the features of the modeling language necessary to represent mechanical elements and mechanical constraints, simulate, analyze and optimize machine systems are discussed. This discussion includes a number of application examples and describes how the constraint approach and modeling language have evolved to provide support throughout the transition from conceptual to embodiment phases and the subsequent detailed design phases.

2. A CONSTRAINT-BASED DESIGN ENVIRONMENT

A constraint-based design environment has been created by the authors with the aim of supporting engineering design per se. The underlying methods and core elements of this environment are summarized in this section and reported in detail elsewhere (Leigh et al., 1989; Mullineux, 2001). The modeling environment incorporates an underlying language based on syntax similar to the C programming language. This language is used to define the system being considered, construct, and manipulate system geometry, and define and solve constraint rules. In addition, the language provides common functions such as array manipulation, numerical differentiation, and general algebraic manipulation.

For the purpose of engineering design, a constraint-based approach is employed to enable the designer to represent what is to be achieved rather than how it is to be achieved (Medland, 1990). These objectives or goals are represented as "constraint rules." These represent the relationships between the design parameters, which must be satisfied if the design is to fulfill all of the requirements. These constraints may represent a variety of performance and physical requirements. In general, it is rare that all the constraint rules are independent. Consequently, they must all be dealt with concurrently and their relationships considered. The aim is therefore to find a solution that satisfies *all* these imposed constraints. It follows that the solution space is the intersection of all the individual constraint fields.

There have been various approaches adopted for the implementation of constraint-based systems and the resolution schemes used to operate with these. The term resolution scheme is used here to denote the underlying reasoning implemented to satisfy the constrained problem. That is to determine an assignment of values to variables that does not violate the imposed constraint set and typically achieves some optimality criterion. The underlying reasoning approaches can be manual, semiautomated, or fully automated. In the case of manual approaches the designer is required to assign values to variables. The effect of the variable assignment(s) is then propagated through the constraint network (O'Sullivan, 2002*b*). Semiautomated approaches such as that presented by Parunak et al. (1999) generally utilize local heuristics and human decision makers to converge on a solution. In contrast to the manual and semiautomated approaches, automated strategies implement a range of numerical techniques to handle the entire value assignment and, either iteratively or through logical reasoning, converge on a satisfactory, optimal, or best compromise solution. Where automated strategies are considered a variety of approaches have been proposed. Some (Bowen & Bahler, 1992; Tay et al., 1997) are based on logic languages such as PROLOG. These tend to require that the design parameters are essentially discrete valued, and hence, can limit their range of application. Systems such as ICAD (KTI, 2003) are similarly formed as an implementation of LISP-like languages. This can mean that the imposed constraints need to be capable of being reordered to allow solutions to be identified.

It is arguable that a more natural approach for engineering and design tasks is one in which the constraints are applied to design parameters that are continuously varying

https://doi.org/10.1017/S0890060406060239 Published online by Cambridge University Press

values. These parameters may represent the geometric attributes of components, their mass properties, kinematic values, and a variety of other performance attributes. In these cases, constraints define relations between the parameters that are either equality or inequality relations, or relations between geometric entities (discussed in Section 5.3). One approach to solving such sets of constraints is again one of reordering to a canonical form that can be solved (Anderl & Mendgen, 1996). This may limit the types of applications that can be handled and the types of constraints that can be applied. It can also mean that cases of multiple feasible solutions (or even no solution) are not easily dealt with. In contrast, the approach taken with the constraint modeling software discussed here makes no large supposition about the form of the engineering constraints except that it is assumed that the underlying variables are (more or less[1]) continuously varying. This means that different forms of constraints between the design parameters can, in principle, be dealt with. This is particularly important for mechanism assembly (Sections 3.2 and 5.3) where a large number of constraints are nonlinear due to the involvement of trigonometric terms relating to angles.

The approach adopted is a general purpose resolution strategy based upon optimization, with the constraints themselves being treated as penalty functions. Each of the constraint rules is effectively created as an expression between the design parameters, which is zero when it is true. Thus, its (absolute) value is a measure of its falseness. A number of numerical techniques are available for the optimization problem. However, as the derivative of the constraint expressions is not immediately available (indeed they may be discontinuous), a direct search approach is preferred. One possibility is the use of genetic algorithms or simulated annealing (Thornton & Johnson, 1996). However, it has been found that these require a large number of function evaluations, and the more "traditional methods" such as Hooke and Jeeves, and Powell's direct search method (Walsh, 1975; Fletcher, 2000; Kolda et al., 2003) work well for the sorts of machine design problems commonly encountered.

A disadvantage of the optimization approach is the possibility that configurations corresponding to false minima may be found. However, it is intended that the user can interact with the system to avoid such problems, and it is felt that the optimization approach has some advantages. In particular, it can identify "best compromise" solutions even in the case when the imposed constraints are in conflict. This can help the designer identify the corresponding conflicts in the physical system, and hence use his/her skill to decide which constraints can be relaxed with safety. In practice, when there are conflicts in the constraint set, the "best compromise" may result in certain physical laws being violated. In such cases it is up to the designer to ensure that these important constraints are not violated. This can be achieved by adjusting the relative weighting applied to individual constraints or through relaxation of one or more conflicting constraints.

The underlying language of the constraint modeler is set up so that the user can declare design variables. These can be of several types, including structured types to represent, for instance, geometric objects. The language supports user-defined functions, which, as in general programming languages, are essentially collections of commands that can be invoked when required. Input variables can be passed into a function and the function itself can return a single value or a sequence of values. The important extension to the function implementation is the use of the "rule" command. Each rule command is associated with a constraint expression between design variables, which is zero (as a real number) when true. A nonzero value is a measure of its falseness. When the function is invoked the constraint expression for each rule command is evaluated and the sum of the squares of these is found. An advantage of using the sum of the squares is that the constraint set is considered in its entirety and there is no explicit requirement for constraint propagation. If this is already zero, then each constraint expression represents a true state. If the sum is nonzero then the system continues with the processing. Within the statement of the function the user specifies, via a command called "var," which design parameters can be varied during a search to make the constraint expressions true. If a minimum of zero can be found then the constraints are fully satisfied. If not, then the minimum represents some form of best compromise for a set of constraints in which there is conflict. Furthermore, it is possible to identify those constraints that are not satisfied, and as a result, less important constraints can be relaxed enabling an overall solution to be determined. In addition to this, particular constraints can be weighted depending on importance. The selection of a suitable weighting must be made by the designer, although the effect of different weightings on the design solution can be easily investigated.

The aim is to provide a single set of values of the design parameters that minimizes the sum. Frequently, there may be multiple solutions for a given system, and hence, the environment is regarded as a tool with which a user interacts. What has been found to happen in practice is that the user can easily identify if the solution (or best compromise) presented is indeed acceptable. If it is not then addition of further constraints is often called for and it is straightforward for the user to define these. This process helps to add to the user's understanding of the design problem. The environment does not therefore explicitly derive design constraints; rather, it supports the user in increasing his/her understanding of the design problem, and hence, elicit the important design constraints.

To illustrate the approach, the solution of a set of linear equations is considered. The problem is defined by the user

[1]The constraint based environment discussed is able to handle continuous variables, discrete variables (lists of values) and bounded variables.

https://doi.org/10.1017/S0890060406060239 Published online by Cambridge University Press

within a simple a function. In the following, there are three linear equations and three unknowns.

```
function solve
{ dec real x, y, z;
  var x, y, z;
  out x, y, z;
  rule( x + y + z − 3);
  rule( 2 ∗ x + y − z − 2);
  rule( x − y + z − 1);
}
```

If this function is invoked directly, then the system varies each of x, y, z to search for a solution and finds that these should all be equal to unity. Suppose that x is removed from the var list (or equivalently if x is defined to be fixed) and is set to be zero. Applying the function again now means that the problem is over constrained. The system finds the best compromise solution which in this case is $y = 1.75$ and $z = 1.25$.

The approach used allows the system to be able to evaluate the constraint expressions, and hence their sum of squares. It does not directly evaluate any derivatives (and indeed these may not exist). Any form of direct search optimization can be implemented to search for a minimum value. The well-known direct search methods due to Hooke and Jeeves and to Powell (Walsh, 1975; Fletcher, 2000; Kolda et al., 2003) have been found to work well, and are currently the techniques implemented within the environment. For the majority of applications, the number of degrees of freedom has seldom needed to exceed 20 or 30, which is ideal for these methods.

3. A CONSTRAINT-BASED DESIGN ENVIRONMENT FOR HIGH SPEED MACHINERY

For the design of high speed machinery and, in particular, systems consisting of a variety of mechanism and linkage assemblies, there are a number of important design tasks. These tasks involve laying out configurations, simulating the operation and evaluating performance. The application of computer-based techniques to this process can considerably reduce time and effort. As a consequence, the designer can consider, compare, and refine a greater number of layouts and, as a result, develop improved design solutions. These design tasks give rise to four corresponding modeling activities: visualizing the system, constructing and assembling system models, simulating and analyzing system performance, and optimization. These four important activities and the means by which they may be supported within a constraint-based design environment are described in the following sections.

3.1. Representing and visualizing system geometry

As has been previously stated, representing and visualizing potential design solutions is an important activity in the design of products and systems. This is particularly the case when considering machine systems that generate complex motions. In such cases, it is necessary to represent the geometry and relative movement of each component during its operation in order for the designer to understand and investigate the large number of highly coupled interactions. As with the majority of design scenarios, both the problem and design solution are likely to be ill defined at the outset. Thus, initially the system geometry may consist only of simple geometric elements, defined by perhaps a length. As the design proceeds, system geometry is defined in more detail, and may eventually involve complex cross sections and intricate profiles. It is therefore necessary for the design environment to provide a means for representing all levels of detail. To support this, the modeling environment allows the construction of parametrically defined two- and three-dimensional geometric entities. Two-dimensional elements are represented as simple wire-frame graphics such as line segments and circular arcs. Three-dimensional elements are represented as solid objects constructed using the ACIS modeler (Corney, 1997), which has been integrated with the software. Both two- and three-dimensional entities are specified in the underlying modeling language. Figure 2 shows a number of examples of two- and three-dimensional mechanisms constructed within the design environment. The figure also shows the corresponding construct used within the underlying language to create the geometry.

As well as representing geometry it is also necessary to provide the means to manipulate geometry. To achieve this, the concept of "model spaces" is adopted (Leigh et al., 1989). Effectively, such a space represents a transform matrix, which maps entities within the space to the world. Model spaces can be embedded one within another to form a hierarchy that is a tree structure with the world space as the "root." In this case, an entity within any space is mapped by each of the transforms between it and the world. An example of a model space hierarchy is shown in Figure 3. This corresponds to the four-bar linkage shown in Figure 4. Within the modeling environment model spaces are defined and associated with each other within the code. For example, in Figure 2, two-dimensional and three-dimensional model spaces are defined in the underlying language using the following construct:

$$\mathrm{m2} = \mathrm{mod2}(0, 0, -20, \mathrm{m1}).$$

Here m2 is embedded in m1. The model space concept is used to aid modeling, and in particular, represent physical connections that are fixed, that is, cannot be violated. The use of model spaces can also act to reduce the degrees of freedom. For example, the connections of a four bar, such as that shown in Figure 4, could be represented by four

https://doi.org/10.1017/S0890060406060239 Published online by Cambridge University Press

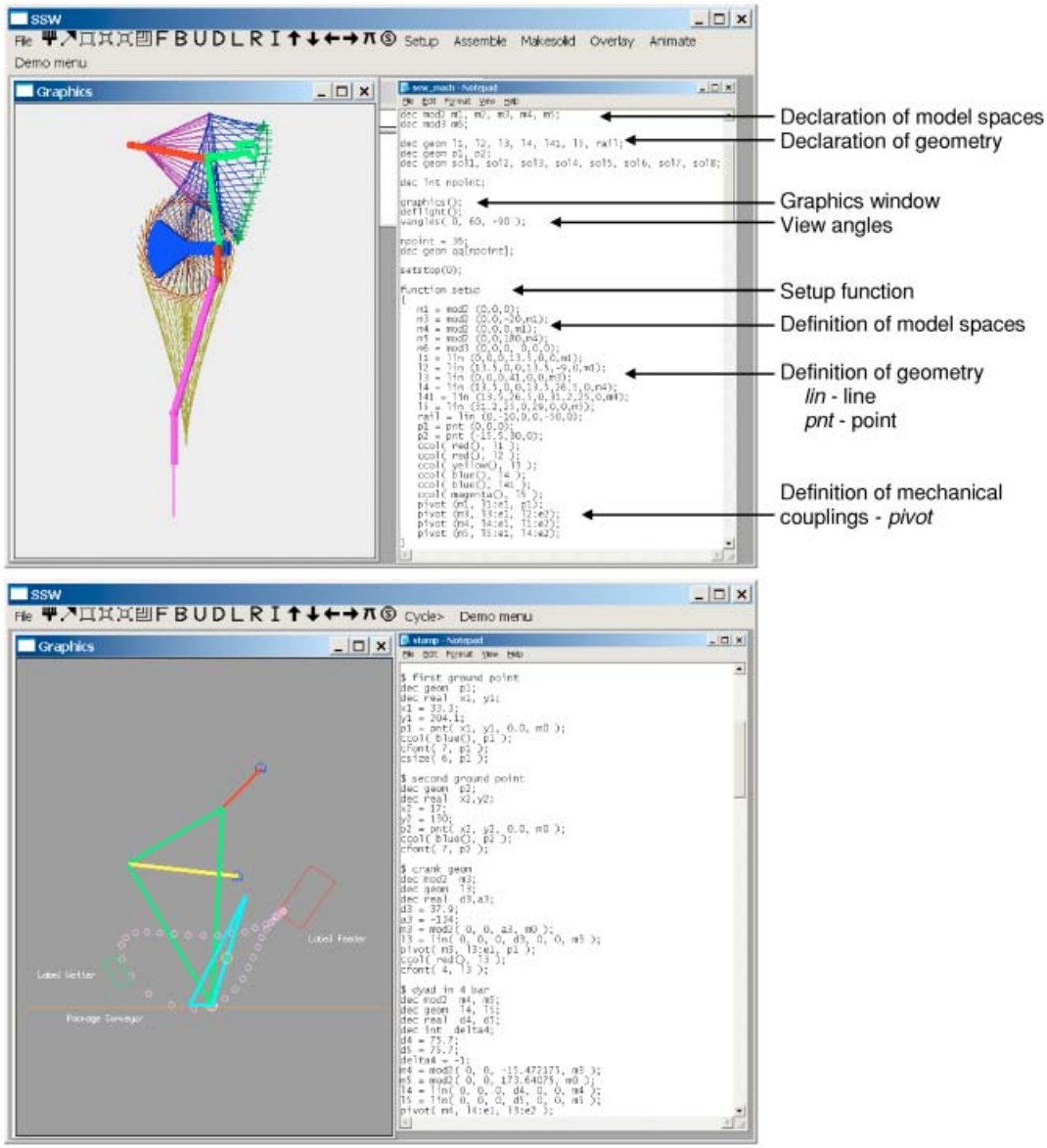

Fig. 2. Models of mechanisms and corresponding macrolanguage to construct geometry. [A color version of this figure can be viewed online at www.journals.cambridge.org]

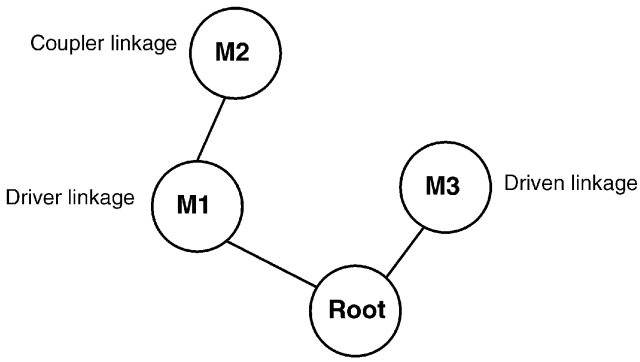

Fig. 3. A model space hierarchy.

constraint rules (Section 3.2). However, in practice, the ground points of the driver and driven link can be embedded in world space, while one end of the coupler can be embedded in the model space of the driver linkage. Hence, only one constraint rule is needed and 2 degrees of freedom: rotation of coupler and driven linkages.

3.2. Constructing and assembling a constraint-based model

Some existing modeling environments use predetermined numerical formulations to simulate the operation of mechanisms. In contrast, the approach taken within the modeling

https://doi.org/10.1017/S0890060406060239 Published online by Cambridge University Press

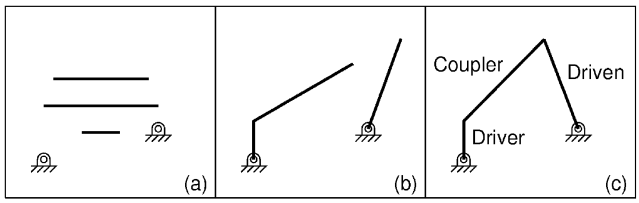

Fig. 4. The construction of a four bar.

environment reported in this paper is to use constraint rules to define important relationships between the geometric elements. It is the use of these constraint rules and their subsequent numerical evaluation that allows the designer to explore and understand the design space, and optimize the design solution within a single environment. Such support is particularly important in the design of mechanisms and is not readily available through other forms of representation. In order to aid the explanation of the constraint rules and constraint resolution techniques a simple example is considered.

This example is a four-bar linkage, which is shown in Figure 4. Initially, the two fixed pivot points are declared and the lines representing the three moving links are defined, each in its own local space (Fig. 4a). The space of the coupler link is "embedded" in the space of the crank, and the spaces for the crank and the driven links are embedded in world space. Then transformations are applied to the links in each respective space and a partial assembly is achieved with the coupler attached to the crank. This is shown in Figure 4b (where rotations have also been applied to aid clarity). If the space of either the crank or the coupler is rotated, the hierarchy of their spaces ensures their ends remain attached.

To complete the assembly, the ends of the coupler and driven link have to be brought together by a constraint rule. Constraint rules define the relationships that need to be satisfied. In the case of the mechanism considered, a constraint rule is applied to describe the desired relationship between the ends of the coupler link and driven link. An attempt is then made to resolve the specified constraints. This involves altering some of the design parameters to satisfy the constraint rules. For the example considered, these design parameters are the angle of rotation of the two model spaces for the coupler and driver linkages.

In the example of the four-bar mechanism, when the constraint rule is applied, it is successfully resolved, and the correct assembly is obtained as shown in Figure 4c. It is important to note that it would be impossible to define the mechanism by purely model spaces alone, as these are hierarchical and the coupler link could not be embedded in two separate hierarchies.

3.3. Simulating operation and analyzing system performance

One of the most important tasks for the design of mechanisms is the analysis of kinematic and dynamic properties. Central to this analysis is the ability to evaluate output motions and perform computations based on the geometric entities and their relative motions with respect to time. In order to simulate the operation of the system, successive time steps are analyzed. For each step the "driven" geometry is incremented and the system assembled. For the mechanism considered in Figure 4 the rotation of the space of the crank linkage is incremented and the other two links assembled. In this manner a simulation of the operation is achieved, as in Figure 5a. Furthermore, if solid objects representing the linkages are created, these can be added to each model space as shown in Figure 5b. They then move correctly when motion is simulated.

In order to support kinematic analysis and further manipulation and assessment of the system properties, the functionality of the modeling environment is used. This enables velocities, accelerations, and jerk to be computed, and these can be recorded and displayed. Velocity, acceleration, and jerk are derived from the displacement by numerical evaluation of the first, second, or third derivatives using values recorded in an array. An example of the output motion of a packaging machine is shown in Figure 6. The figure also shows the velocity and acceleration characteristics.

3.4. Optimizing system performance

As discussed in the previous section, the modeling environment described uses minimization techniques to determine a solution that satisfies or best satisfies the given set of constraint rules. The same search algorithms and rule constructs used to set up and satisfy the constrained problem can also be used to define objective functions and determine an optimized solution. For the purpose of designing mechanism and machine systems, the methodology for system optimization involves incorporating the system assembly and system simulation procedures described in Sections 3.2 and 3.3, respectively, into an iterative optimization loop. In a manner similar to the system assembly process, design parameters are specified and rules constructed. These rules may represent any aspect of system performance or system properties that can be represented

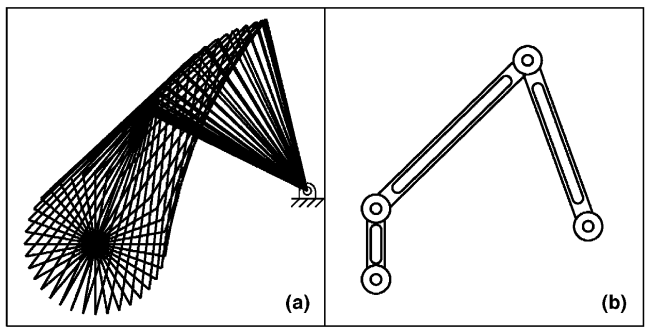

Fig. 5. Simulating the operation of a four bar.

https://doi.org/10.1017/S0890060406060239 Published online by Cambridge University Press

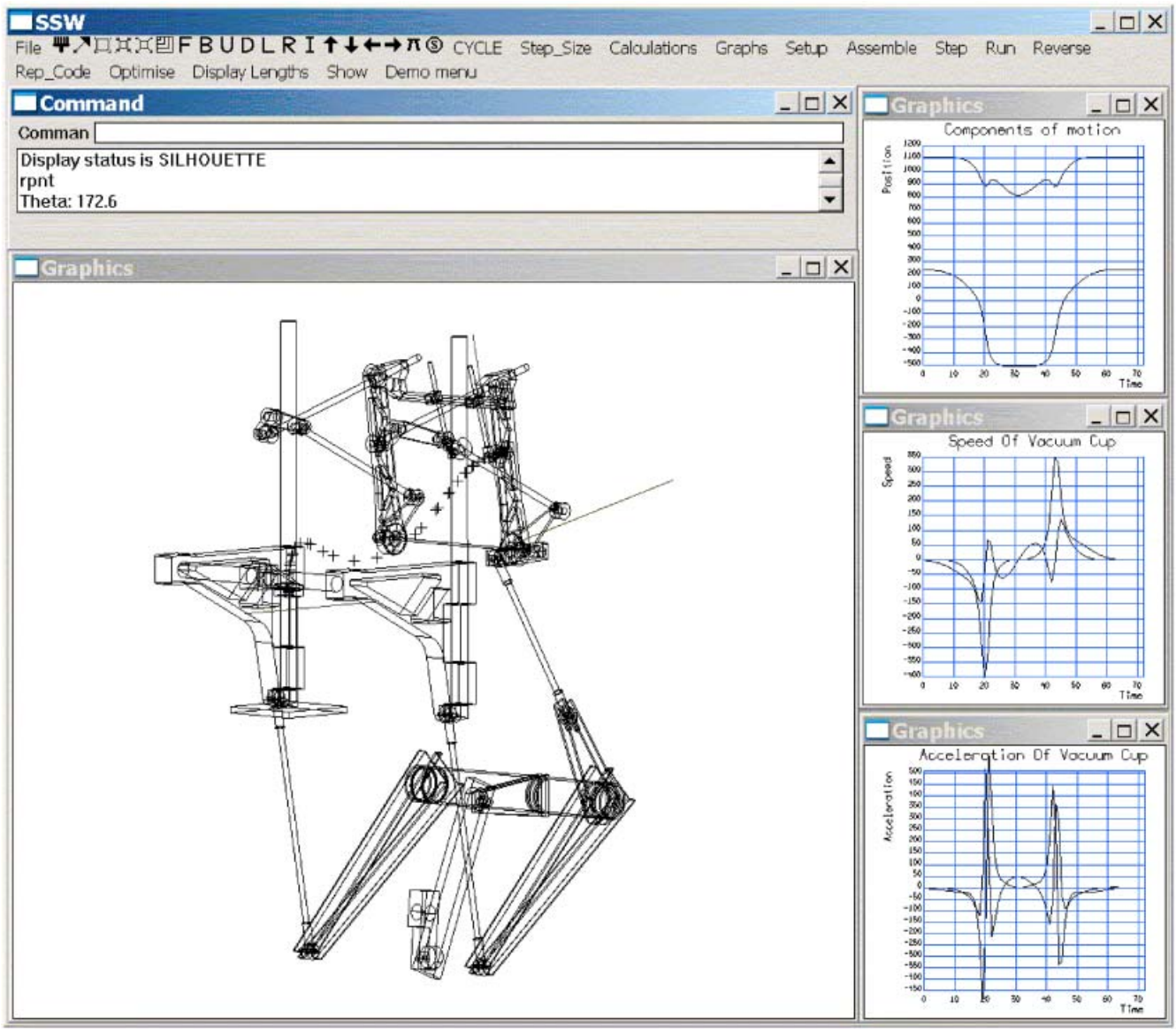

Fig. 6. A kinematic analysis of a complex packaging machine. [A color version of this figure can be viewed online at www.journals.cambridge.org]

algebraically using equality or inequality relations. The goal is to determine a configuration where the value of these rules is minimized. The overall process therefore involves varying a set of design parameters (optimization parameters), incrementing the system model through its operational cycle, solving the assembly constraints at each step using a second set of design parameters (assembly parameters), evaluating the objective function(s) (rules), and then specifying an altered set of design parameters (optimization parameters). It is important to note that although the construct of the rules for optimization and assembly are the same, the two sets of design parameters (assembly parameters and optimization parameters) are separate and distinct. For the purpose of designing mechanisms, the optimization parameters may include linkage lengths, pivot points, and cam profiles. The optimization process is terminated when either an optimized solution is found, a best compromise is determined, or after a predetermined number of iterations.

4. REPRESENTING MECHANICAL ELEMENTS

In order to represent mechanism and linkage systems, three fundamental mechanical elements need to be considered. These are linkage elements, cams, and motion paths. These components and the means by which they are constructed within the modeling environment are summarized in the following sections.

https://doi.org/10.1017/S0890060406060239 Published online by Cambridge University Press

4.1. Linkages

Within the modeling environment, mechanical links can be represented as both two-dimensional and three-dimensional geometric entities. However, it is the two-dimensional entities that are generally used to construct the constraint-based model. This corresponds to the "stick diagram" often used in practice to express the key features of a mechanical assembly and its operation. The two-dimensional representations are defined as Euclidean line segments, which are specified by two end points in three-dimensional space. Examples of the lin construct are highlighted in Figure 2. It is these end points that are used to define constrained relationships between the mechanical elements. Within the modeling language, each line segment is assigned a unique label and specified with an initial length. In general, the line segments (linkages) are also associated with a particular model space, thus allowing translations, rotations, and scaling to be applied. It is also possible to parametrically define the line segments. This is particularly useful during optimization as it enables the dimensions of individual links to be altered.

4.2. Cams and motion paths

In addition to mechanical links, it is also necessary to represent cams, both rotational and linear, and motion paths. To do this, the modeling environment allows the definition of free-form curves, both open and closed (McGarva & Mullineux, 1993). The curve has the form of a B-spline and can be either a planar or space curve. The curve is defined by a set of controls points specified via the modeling language and represents a geometric entity. In addition, a curve can be fitted through a number of prescribed precision points. Curves can be manipulated in a similar manner to a mechanical link. For example, closed curves representing cams can be rotated about a central point in order to simulate operation. Furthermore, similar forms of constraint rules can be applied to specify relationships between the corresponding mechanical elements.

5. REPRESENTING MECHANICAL CONSTRAINTS

For the purpose of representing mechanism and linkage-based machine systems, there are three fundamental mechanical constraints. These constraints effectively define two classes of mechanical coupling and "assembly constraint." Within the modeling environment, rotational and sliding joints are represented by direct manipulation of the model space hierarchy. These mechanical couplings are therefore fixed. That is to say they are not considered within the constraint satisfaction process.

5.1. Rotational joint

This class of coupling occurs between the ends of two linkages or between a linkage and what can be thought of as a ground point. Because of the prevalence of this class of joint within mechanism systems, a built-in function has been defined within the language. The "pivot" function has the effect of coupling two model spaces (and their associated geometry) such that joint positions are brought together and subsequently only rotational motion is permitted. In order to achieve this, the relative center of rotation needs to be defined. Typically, this involves the end point of one or more linkages (line segments) and a point in world space (defined by its coordinates in three-dimensional space).

5.2. Sliding joint

The second class of mechanical coupling involves a sliding joint. Here, a linkage is allowed to slide along a predefined linear path. Again, the modeling environment has a built-in function to define and solve this class of mechanical coupling. The "slid" command ensures that the appropriate end of a linkage embedded in a particular model space always lies on a linear path associated with another model space.

5.3. Assembly constraint

For the purpose of using constraints to represent mechanism assembly and operation, one or more "assembly constraint" needs to be applied to complete the definition. It is not possible to fully constrain a mechanism using pivot joints and sliding joints alone. An initial assembly can be created using these couplings to define hierarchical relationships between model spaces. The tree form of the hierarchy, shown in Figure 3, is insufficient to describe the full assembly as this requires loops within the structure. To deal with this, final assembly constraints are set up, often using a binary function called "on." These geometric entities include the end points of line segments, the lines themselves, or other entities, including open and closed curves and points in space. When the "on" function gives a zero value, the two entities are coincident.

If the example presented in Figure 4 is considered, an assembly constraint is specified between the end of the coupler link and the driven link. In the user language, the constraint rule is expressed as follows.

```
rule( l2:e2 on l3:e2 );
```

here, l2 represents the line of the coupler and l3 represents that of the driver. The construction comprising a colon followed by e2 is used to denote the second end point of the line in question. During constraint resolution an attempt is made to minimize the value of this constraint, so that the distance between the ends of the two lines becomes zero and the lines come together.

6. USING CONSTRAINT-BASED MODELS FOR DESIGN

In the machinery design sector, the designer is generally faced with two classes of problems. These involve either

https://doi.org/10.1017/S0890060406060239 Published online by Cambridge University Press

redesigning an existing machine for an altered set of performance requirements, or the creation of new machine assemblies to meet a different, sometimes totally new, set of requirements. The flexibility of the constraint-based approach means that both classes of design problems can be supported by the design environment.

In particular, existing machine configurations (including component sizes) can be modeled and their performances evaluated (Hicks et al., 2002*a*). Similarly variant or new machine configurations can be modeled, their performance requirements specified, and component sizes determined. For example, if a cam and linkage assembly is considered, either the cam profile can be specified and the resulting motion determined, or the required motion can be specified and the cam profile necessary to deliver the motion determined. This ability to evaluate the performance of a design or to carry out "design by performance" is enabled by the bidirectional properties of the constraint rules. For the cases considered in Figure 7, the model can be used for either design scenario by simply transferring an assembly constraint from the cam profile and the follower to the final driven linkage and the path representing the desired output motion. For example, in Figure 7a, the cam profile for a new machine is generated by considering the desired motion. An existing system is redesigned in Figure 7b. The upper portion highlights the current output motion, desired output motion and current cam profile. In the lower portion a modified cam profile is created to achieve the desired motion.

6.1. Designing large-scale complex systems

Thus far, the examples presented have included predominantly what can be thought of as assembly constraints. In practice, there are also many physical and performance constraints that need to be considered. Such constraints may govern velocities, accelerations, relative positions, size restrictions, and systems properties. In order to consider these constraints during system design, additional constraints rules can either be incorporated into the set of assembly constraints (Section 3.2) or specified within an optimization function (Section 3.4). Both assembly constraints and objective functions are defined using the *rule* structure discussed in Section 5.3.

It is usually the case that both the design of the assemblies and the various design constraints evolve as the design proceeds. The application of constraint-based approaches using minimization techniques is particularly suited to this class of problems. This is because the constraints are solved collectively and do not rely upon any particular ordering. Further design parameters and constraint rules can be included and excluded at any time. However, simulation involving hundreds of design parameters has been shown to be numerically time consuming (Molenbroek & Medland, 2000). In practice, the scale issues can often be overcome through an appropriate resolution strategy. This strategy may involve identifying groups of related design parameters and/or identifying critical design parameters. The use of different resolution strategies is illustrated by three case studies. These are shown in Figure 8a–c. These are now described and for each case the numbers of design parameters and constraint rules is given. Part a shows a carton erection machine. Here there are two assemblies, 31 linkages, 16 design parameters, and 15 constraint rules. In this case, all linkages are mechanically coupled to a single drive. Hence, the assemblies are treated as being dependent and all the constraint rules are solved collectively. For this case, simulation takes of the order of 120 s using a PC with a 1-GHz processor. This involves 20 time steps and approximately 20 iterations for each step. For the second case considered, the machine system involves four cam and linkage assemblies. For the purpose of simulating the operation these assemblies can be resolved independently for every time step. In total this model involves more than 20 design parameters and 16 constraint rules. The third case involves a constraint-based mannequin. The model of the mannequin involves 19 linkages, 86 design parameters, and 27 constraint rules. These rules govern not only mechanical couplings and relationships but also system properties such as balance. Here, sensitivity analysis is first undertaken to establish the critical design parameters. Sensitivity analysis involves exhaustively perturbing each design parameter and evaluating its impact upon system performance. The most critical design parameters are then used for constraint satisfaction. In the case of the mannequin, sensitivity analysis can reduce the number of design parameters to only 30. Furthermore, research has shown that the solutions determined with only a subset of the design parameters show a strong correlation to those observed during practical studies of human movement (Molenbroek & Medland, 2000).

6.2. Evaluating design solutions

The constraint model can be used to support not only the generation of design solutions that satisfy particular requirements but also may be used for the purpose of testing design alternatives or functional structures. The evolving set of constraint rules provides a means against which a given solution can be tested throughout the process. For example, a previous design solution can be tested against the constraint set. The constraints that are violated are highlighted and a measure of violation is given. This measure is equal to the error term evaluated during constraint satisfaction (Section 2). This provides a means by which possible design solutions can be continually tested throughout the various tasks of conceptual design and embodiment design. For example, in the design of a family of products, such as a bicycle, it is particularly useful to be able to evaluate the impact of changes in standard components across the entire range, and in particular, different rider groups. For such a design scenario, a constraint-based model has been constructed that considers the bicycle and the rider (Hicks et al., 2005*b*). Here the designer is provided with feedback on

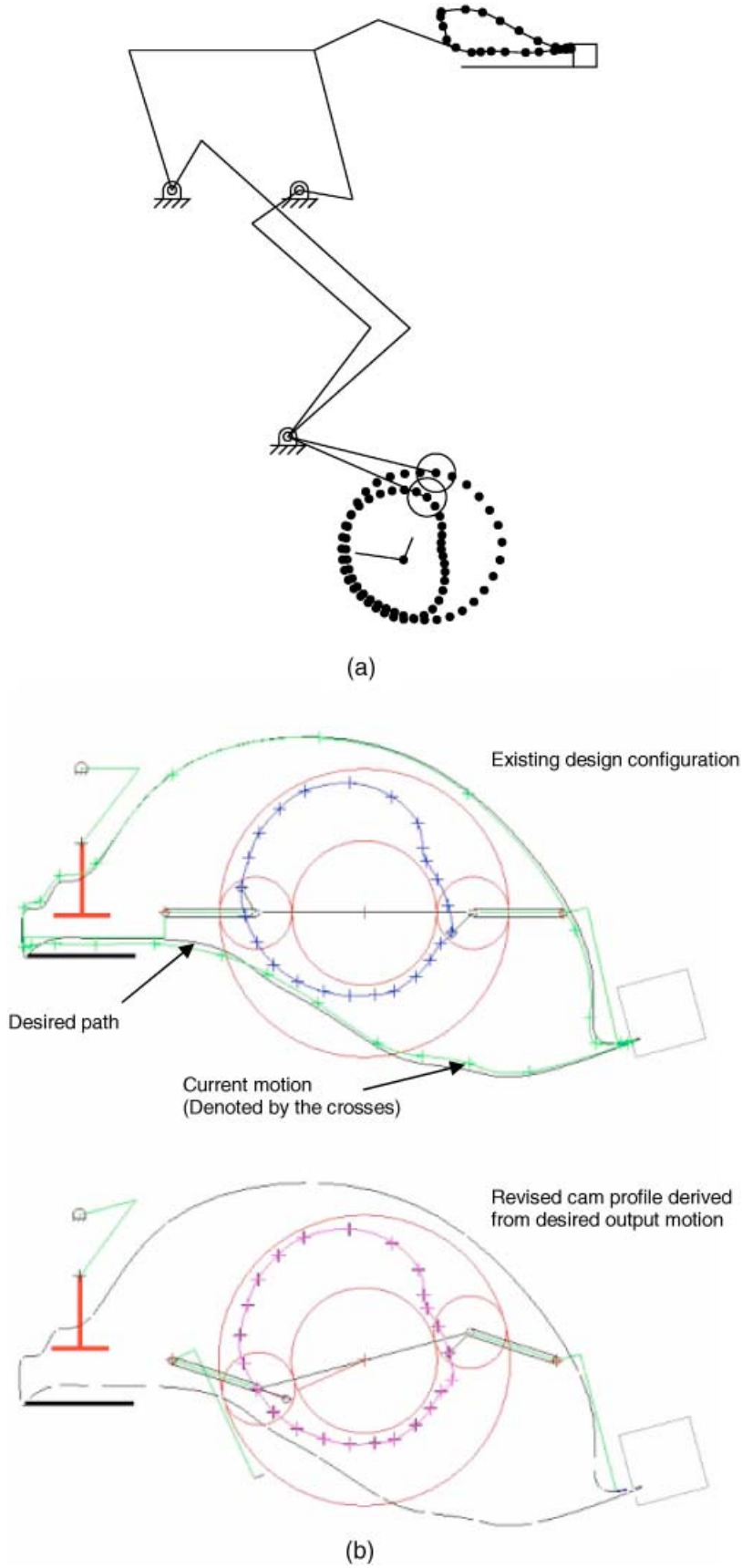

Fig. 7. Designing by performance for cam and linkage assemblies. [A color version of this figure can be viewed online at www.journals.cambridge.org]

https://doi.org/10.1017/S0890060406060239 Published online by Cambridge University Press

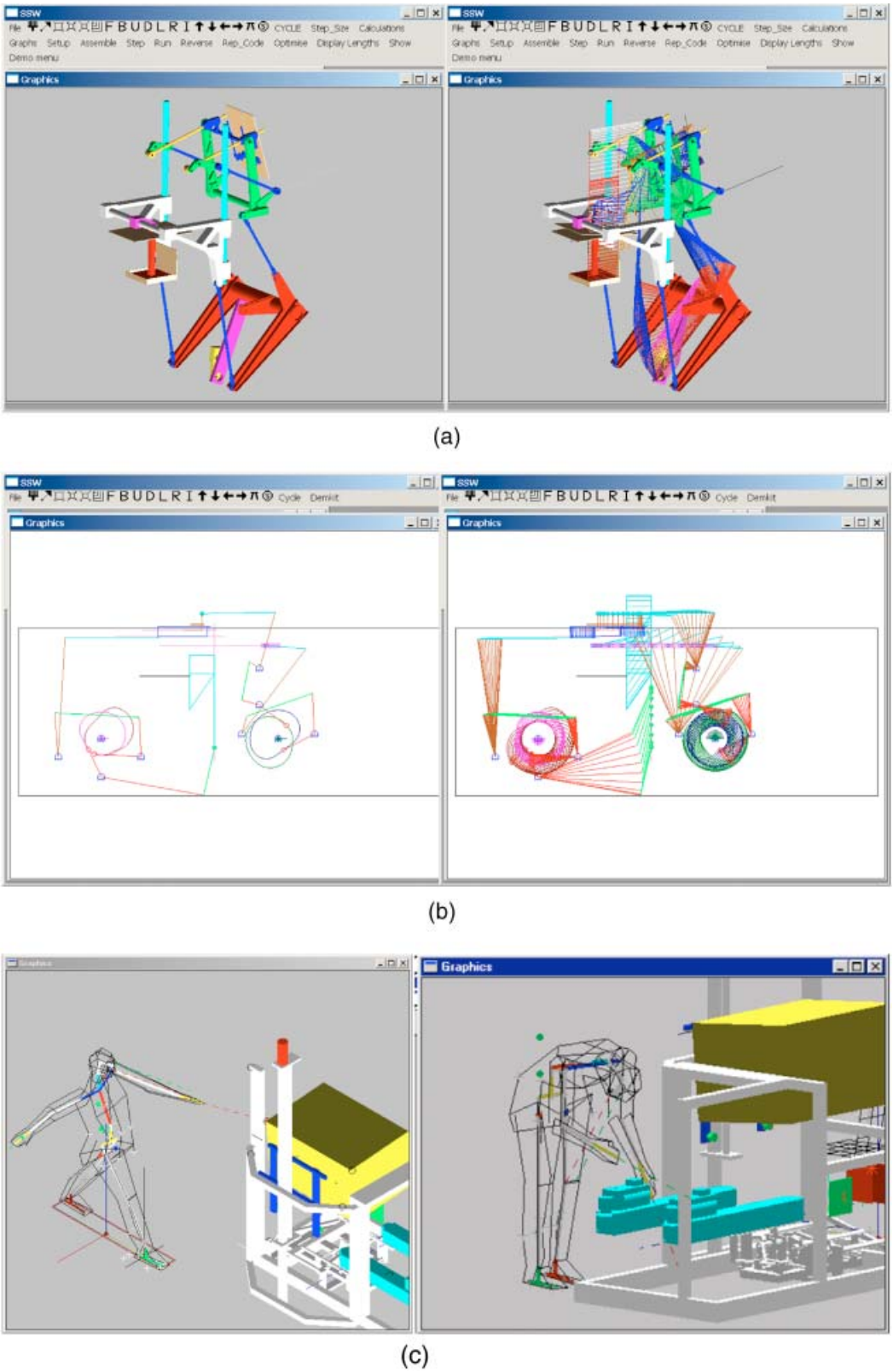

Fig. 8. Constraint-based models of complex systems. [A color version of this figure can be viewed online at www.journals.cambridge.org]

https://doi.org/10.1017/S0890060406060239 Published online by Cambridge University Press

those rules that are no longer satisfied, and importantly, is provided with a measure of the violation. This enables different configurations to be tested and the effect of changes to be evaluated.

In addition to testing design solutions statically through evaluation of the constraint rules, it is also possible to evaluate designs throughout the operational cycle. This is particularly important in the design of mechanism and machine systems where it is important that transmission angles are within certain limits and those linkages do not fail. Furthermore, the inclusion of solid objects enables interference checking (collision detection) to be performed over the operational cycle. Example of the use of collision detection in the design of a product feed system and an ejection mechanism for a machining center are shown in Figure 9. Here the design of the mechanism and also the profile of the "scoop" are considered in order to determine an optimum solution that does not interfere with the tooling or machined component.

6.3. Optimizing design solutions

In the design of mechanism and machine systems there is frequently a need to achieve a desired motion profile and satisfy particular kinematic requirements. Such requirements may govern the entire output motion or only a small

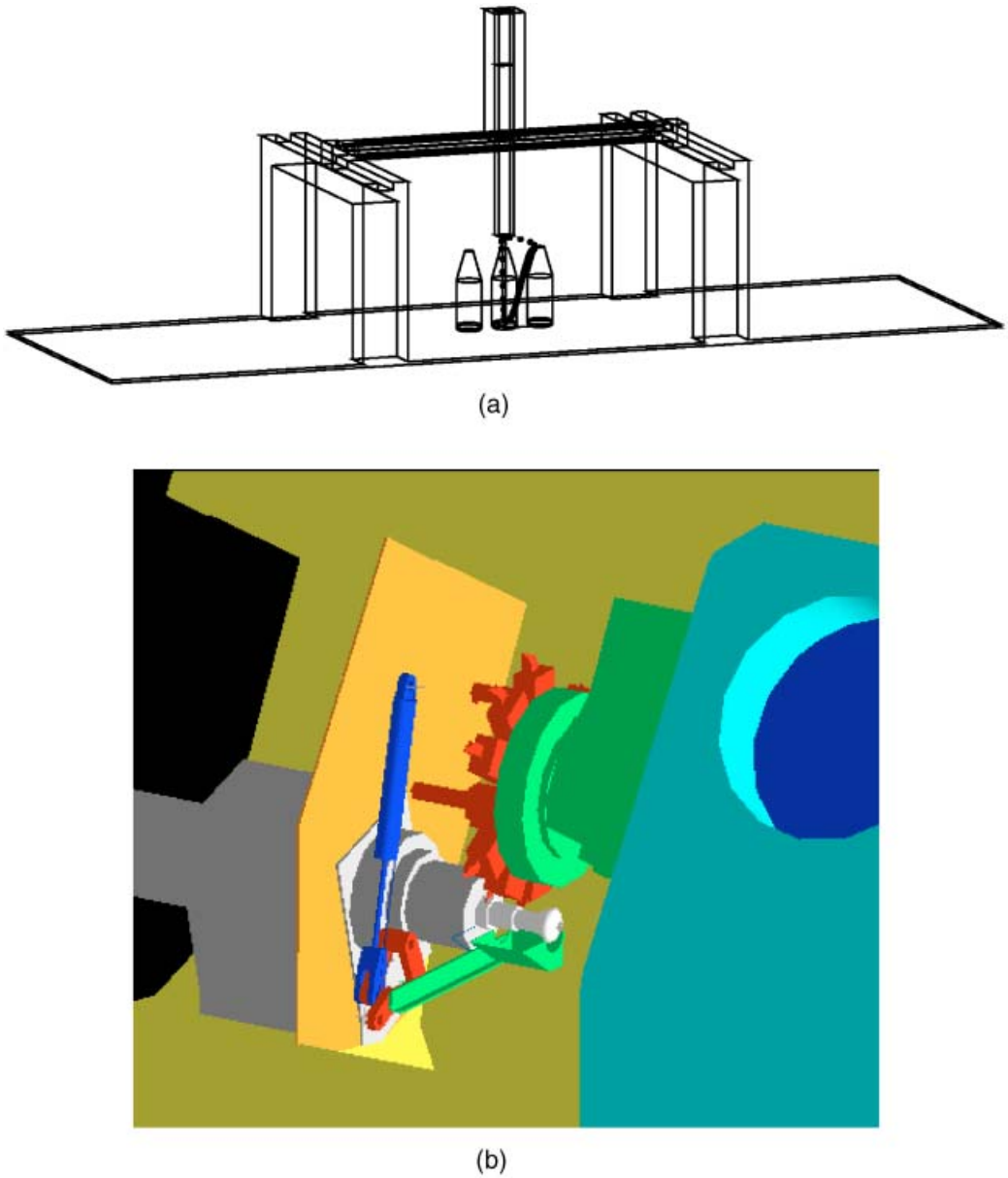

Fig. 9. The incorporation of interference checking in systems design. [A color version of this figure can be viewed online at www.journals.cambridge.org]

https://doi.org/10.1017/S0890060406060239 Published online by Cambridge University Press

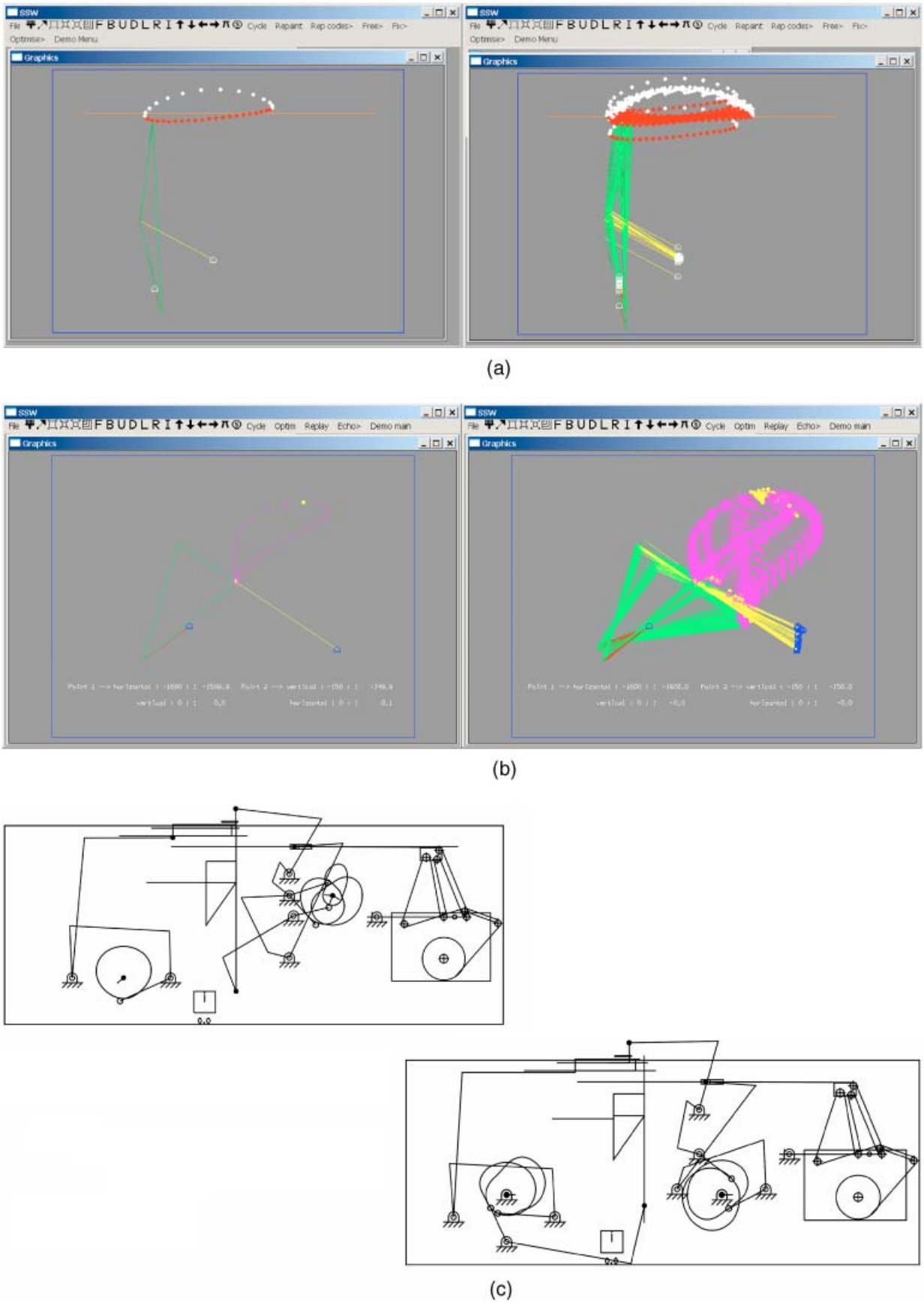

Fig. 10. Examples of optimized machine systems. [A color version of this figure can be viewed online at www.journals.cambridge.org]

https://doi.org/10.1017/S0890060406060239 Published online by Cambridge University Press

portion of that motion. One approach to achieving this is to reverse engineer the design solution using a path that represents the desired output motion, as suggested in Section 6. Although such an approach is suitable for design scenarios involving redesign or refinement of existing systems, it is less suitable where new designs are considered. This is because simply reverse engineering the design solution often results in unsuitable cam profiles, linkage assemblies with excessive transmission angles, or unnecessary large mechanisms. To overcome this, additional constraint rules and associated optimization functions are required. Typically, the constraint rules either define limits that must not be exceeded or terms to be minimized. For example, the limits may often be specified for physical size, allowable transmission angles and cam gradients, and it is often desirable to reduce maximum velocity, acceleration, and jerk. To illustrate the importance of constraint optimization in the design of mechanisms, three examples are shown in Figure 10. For each example the initial design configuration and optimized design configuration are shown and the constraints used for optimization are summarized as follows:

- case a: mechanism geometry and ground points are optimized to achieve an almost horizontal stroke for a portion of the cycle;
- case b: mechanism geometry and ground points are optimized to match velocity at two points in the cycle; and
- case c: cam and linkage assemblies are optimized to reduce maximum accelerations and jerk.

7. CONCLUSIONS

The use of constraint-based techniques in the context of design has been evolving for almost three decades. Many techniques have been developed and applied to the different stages of the design and manufacturing processes with varying degrees of success. Furthermore, many constraint approaches and techniques to enable their application to more general industrial problems are still the subject of ongoing research. In contrast, this paper describes a constraint-based design environment that has been largely developed through application to industrial problems. The aim of the design environment is to support engineering design per se, and in particular, system embodiment, detailed design, and optimization.

This paper provides an overview of the representation and underlying methods implemented in the constraint-based environment. These include the "rule" construct and the use of optimization methods to collectively satisfy the entire constraint set. The functionality and capabilities of the design environment are discussed with particular reference to the use of constraints for design. More specifically, the core elements of the design environment that support the visualization of system models, the construction and assembly of constraint-based models, the simulation and analysis of system performance, and optimization of the design solution are described. The paper also discusses the underlying elements of the modeling language that are necessary to represent and handle the full range of mechanical elements and mechanical constraints present within mechanism and machine systems.

Following the detailed overview of the design environment, the use of constraint-based models for design tasks associated with mechanism systems and high speed machinery is discussed. This discussion deals with three important elements: handling evolving design knowledge, evaluating design solutions, and system optimization. A range of case study examples is included to illustrate the capability of the environment and its ability to support the various tasks of design. These include the ability to reverse engineer systems to achieve desired motions, evaluate design alternatives, assess the impact of changes, determine an optimized solution, and explore design tradeoffs. Such functionality is ultimately a prerequisite for effectively exploring the design space and supporting the designer in generating a fundamental understanding of the particular design problem.

ACKNOWLEDGMENTS

The work reported in this paper was supported by a number of grants for the Engineering and Physical Sciences Research Council, the Department of Trade and Industry, and the Department for Environment Food and Rural Affairs, involving a large number of industrial collaborators. In particular, current research is being undertaken as part of the EPSRC Innovative Manufacturing Research Centre at the University of Bath (reference GR/R67507/01). The authors gratefully express their thanks for the advice and support of all concerned.

REFERENCES

Anantha, R., Kramer, G.A., & Crawford, R.H. (1996). Assembly modelling by geometric constraint satisfaction. *Computer-Aided Design 28(9)*, 707–722.

Anderl, R., & Mendgen, R. (1996). Modelling with constraints: theoretical foundation and application. *Computer-Aided Design 28(3)*, 155–168.

Armarego, E.J.A., & Ostafiev, D. (2003). Multi-constraint optimization and cutting conditions selection in process turning operations with modern chip breaker tools. *Proceedings of the Institution of Mechanical Engineers*, *Part B: Journal of Engineering Manufacture 217(1)*, 57–71.

Bowen, J., & Bahler, D. (1992). Frames, quantification, perspectives and negotiations in constraint networks in life-cycle engineering. *International Journal for Artificial Intelligence in Engineering 7*, 199–226.

Camlinks Ltd. (2005). Machine and motion design software. Available on-line at http://www.camlinks.com

Cianchi, M. (1988). *Leonardo da Vinci's Machines*. Rome: Becocci Editore.

Corney, J. (1997). *3D Modelling with the ACIS Kernel and Toolkit*. Chichester: Wiley.

Counsell, J., Porter, I., Dawson, D., & Duffy, M. (1999). Schemebuilder: computer aided knowledge based design of mechatronic systems. *Assembly Automation 19(2)*, 129–138.

Fine Systems Corporation Limited. (2005). INca, INBIS constraint analyzer—commercial version of the University of Bath Constraint Modelling Software. Available on-line at http://www.finegroup.co.kr/inbis-software/inca/applications.htm

Fletcher, R. (2000). *Practical Methods of Optimisation*, 2nd ed. New York: Wiley.

Gray, A., & Atalan, H. (2002). Integrating the design process. *3rd Worldwide Aerospace Conf. and Technology Showcase*, Paper No. 2001-61, Toulouse.
Hicks, B.J., Bowler, C., Medland, A.J., & Mullineux, G. (2002*a*). A redesign methodology for analysing and improving the performance capability of packaging machinery. *International Journal of Industrial Engineering 9(4)*, 389–398.
Hicks, B.J., Bowler, C., Medland, A.J., & Mullineux, G. (2002*b*). The role of virtual sequence simulation in the analysis of packaging machines. *Journal of Engineering Design 13(11)*, 19–32.
Hicks, B.J., Culley, S.J., & Mullineux, G. (2005*a*). The representation of engineering systems for their embodiment with standard components. American Society of Mechanical Engineers. *Journal of Mechanical Design 127(3)*, 424–432.
Hicks, B.J., Mullineux, G., & Medland, A.J. (2005*b*). Constraint-aided product design. *Journal Acta Polytechnica 45(3)*, 31–36.
Kolda, T.G., Lewis, R.M., & Torczon, V. (2003). Optimization by direct search: new perspectives on some classical and modern methods. *SIAM Review 45(3)*, 385–482.
KTI. (2003). The ICAD system. Available on-line at http://www.ktiworld.com/our_products/icad.shtml
Leigh, R.D., Medland, A.J., Mullineux, G., & Potts, I.R.B. (1989). Model spaces and their use in mechanism simulation. *Proceedings of the Institution of Mechanical Engineers*, *Part B: Journal of Engineering Manufacture 203(2)*, 167–174.
Mailharro, D. (1998). A classification and constraint-based framework for configuration. *Artificial Intelligence in Design, Analysis and Manufacture 12(5)*, 383–397.
Markus, A., Vancza, J., & Kovacs, A. (2002). Constraint-based process planning in sheet metal bending. *CIRP Annals—Manufacturing Technology 51(1)*, 425–428.
McGarva, J.R., & Mullineux, G. (1993). Harmonic representation of closed curves. *Applied Mathematical Modelling 17(2)*, 213–218.
Medland, A.J. (1990). Whole in One. *Engineering Design Education and Training Summer*, 16–18.
Molenbroek, J.F.M., & Medland, A.J. (2000). The application of constraint processes for the manipulation of human models to address ergonomic design problems. *Proc. 3rd Int. Symp. Tools and Methods of Competitive Engineering*, pp. 827–835.
MSC Software Corporation. (2005). MECHANISM/PRo simulation of kinematic motion. Available on-line at http://www.mscsoftware.com/support/prod_support/mechanism/
MTA. (2005). Basic facts. Annual report of the Manufacturing Technology Association. Available on-line at http://www.mta.org.uk/pdf/basic_facts.pdf
Mullineux, G. (2001). Constraint resolution using optimisation techniques. *Computers & Graphics 25(3)*, 483–492.
O'Sullivan, B. (2002*a*). *Constraint-Aided Conceptual Design*. London: Professional Engineering Publishing Limited.
O'Sullivan, B. (2002*b*). Interactive constraint-aided conceptual design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 16(4)*, 303–328.
Parunak, H.V.D., Ward, A.C., & Sauter, J.A. (1999). The MarCon algorithm: a systematic market approach to distributed constraint problems. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 13(3)*, 217–234.
PTC. (2005). Pro/ENGINEER mechanism design solutions. Available on-line at http://www.ptc.com/products/proe/mdx/
Richards, T.G., Hughes, E.J., & Tilley, D.G. (1999). A multimedia approach to fluid power systems design. *Proc. IEEE Int. Conf. Multimedia Computing and Systems*, pp. 792–796, Florence, Italy.
Schmidt, L.C., Shi, H., & Kerkar, S. (2005). A constraint satisfaction problem approach linking function and grammar-based design generation to assembly. *Journal of Mechanical Design 127(2)*, 424–432.
Shigley, J.E., & Uicker, J.J. (2003). *Theory of Machines and Mechanisms*, 3rd ed. New York: Oxford University Press.
Stumptner, M., Freidrich, G.E., & Haselböck, A. (1998). Generative constraint based configuration of large technical systems. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 12(4)*, 307–320.
Tay, J.C., Huang, S.Y., & Quek, C. (1997). CSL (Part I): modelling general N-ary, logical CSPs, *Proc. Int. Conf. Tools With Artificial Intelligence*, pp. 414–421.
Thornton, A.C., & Johnson, A.L. (1996). CADET: a software support tool for constraint processes in embodiment design. *Research in Engineering Design 8(1)*, 1–13.
Timoshenko, S.P., & Young, D.H. (1948). *Advanced Dynamics*. New York: McGraw–Hill.
UGS Corporation. (2005). NX mechanism Simulation. Available on-line at http://www.ugs.com/products/nx/docs/fs_ideas_mechanism_simulation.pdf
US Census Bureau. (2002). 2002 Economic census manufacturing industry series. packaging machinery manufacturing: 2002. Available on-line at http://www.census.gov/prod/ec02/ec0231i333993.pdf
Visual Components Ltd. (2005). 3D Manufacturing simulation and visualisation software. Available on-line at http://www.visualcomponents.com/portal/company/company
Walsh, G.R. (1975). *Methods of Optimization*. London: Wiley.

Ben Hicks is a Senior Research Fellow in the Department of Mechanical Engineering, University of Bath. His research interests lie in the area of developing tools and methods for the support of engineering design. Dr. Hicks' recent work has been in the design and manufacture of high speed machinery, product modeling and analysis, and machine–material interaction. He is also interested in the development of improved techniques for handling formal and informal engineering information.

Glen Mullineux is a Reader at the University of Bath and has authored or coauthored over 120 publications. He attained his doctorate in mathematics and has been working in the area of geometric modeling and computer-aided design for a number of years. Dr. Mullineux is particularly interested in the application of mathematical techniques in the support of engineering design problems and the modeling and optimization of engineering systems.

Tony Medland is Professor of design engineering at the University of Bath. He has had many years of practical experience in design for companies and as a consultant. His research interests lie in the areas of design and problem solving and how these apply to the overall design process from initial concept through to manufacture, inspection, and beyond. Professor Medland's current work deals with the creation of a constraint-based manikin for investigating the man–machine interface and designing disability aids.