Computation reuse for rigid-body dynamics

Anne Loomis
Computer Science Department
Dartmouth College
Hanover, NH 03755
Email: loomis@cs.dartmouth.edu

Devin Balkcom
Computer Science Department
Dartmouth College
Hanover, NH 03755
Email: devin@cs.dartmouth.edu

***Abstract*— The accelerations of and forces among contacting rigid bodies may be computed by formulating the dynamics equations and contact constraints as a *complementarity problem* [1]. Dantzig's algorithm, when applicable, will find a solution to the linear complementarity problem corresponding to an assembly with n contacts in $O(n)$ *major cycles*.**

Can the dynamics of an assembly be computed more quickly if the dynamics of a subassembly are already known? This paper shows that Dantzig's algorithm will find a solution in $O(n-k)$ major cycles if the algorithm is initialized with a solution to the dynamics problem for a subassembly with k internal contacts.

We apply this observation to two robotics problems: *dynamic simulation* and *assembly sequence planning*. In dynamic simulation, the positions of several bodies might remain fixed during a sequence of frames. We compute the dynamics of this motionless subset (which might not be motionless when considered in isolation), and use the result to initialize the computation for the entire assembly. In assembly planning, non-disjoint sets of objects are typically considered sequentially by the planner. If the configuration of only one body is varied, the dynamics of successive assemblies can be computed in a constant number of major cycles.

I. Introduction

The problem of determining the motion of and forces among contacting rigid bodies is fundamental to many areas of robotics, including dynamic simulation, control, and manipulation planning. The primary contribution of this paper is the observation that in many cases, common physical structure can be exploited to solve a sequence of related dynamics problems more efficiently than if each problem were considered in isolation.

The typical model of contacting rigid bodies consists of the Newton-Euler dynamics equations, unilateral-force constraints, non-penetration constraints, and frictional constraints. With the correct choice of coordinates, the equations and constraints form a *complementarity problem* [1]. We consider only *linear* complementarity problems (LCPs), such as those that arise in planar systems, in spatial systems without friction, and in spatial systems with linearized friction cones.

For some constant $n \times n$ matrix $\mathbf{A}$, and constant vector $\mathbf{b}$ of length n, a *complementary feasible solution* to a linear complementarity problem $(\mathbf{A}, \mathbf{b})$ is a a pair of vectors $(\mathbf{f}, \mathbf{a})$ satisfying

$$\mathbf{a} = \mathbf{A}\mathbf{f} + \mathbf{b}, \tag{1}$$

$$\mathbf{a}, \mathbf{f} \geq 0, \tag{2}$$

$$\mathbf{a}^T \mathbf{f} = 0. \tag{3}$$

The structure of the matrix $\mathbf{A}$ and the physical interpretation of the vectors depend on the particular dynamics formulation. For example, in [2], $\mathbf{a}$ is a vector of relative accelerations at the contact points, $\mathbf{f}$ is a vector of forces applied at the contacts, and

$$\mathbf{A} = \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T \qquad \text{and} \qquad \mathbf{b} = \mathbf{J}\mathbf{M}^{-1}\mathbf{F}^{\text{ext}}, \tag{4}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{J}$ is the Jacobian relating motion of the bodies in generalized coordinates to motion of the contact points, and $\mathbf{F}^{\text{ext}}$ is the vector of external and velocity-dependent forces. In [3], $\mathbf{a}$ is a vector of velocities, and $\mathbf{f}$ contains impulses. In most formulations, $\mathbf{A}$ can be partitioned in such a way that each block corresponds to a particular contact present in the physical system.

The Dantzig algorithm performs a series of n or fewer *major cycles* to successively satisfy each constraint $a_i \geq 0, f_i \geq 0$, as described by equation 2. We show that the solution to the dynamics equations for a structure can be found in $n-k$ or fewer major cycles of Dantzig's algorithm if the solution for the dynamics equations are known for any k-dimensional subproblem. Although this result seems physically intuitive, it is not clear how computation can be similarly reused in other complementarity algorithms such as Lemke [4] or PATH [5].

We consider two example applications of this algorithm: dynamic simulation, and assembly planning.

- **Dynamic simulation.** During simulation, the positions of several bodies might remain fixed throughout a sequence of frames; figure 1 shows an example. Our algorithm computes the dynamics to identify the set of motionless bodies, re-computes the dynamics of that substructure, and uses the result as a starting point to compute the dynamics of the complete structure in successive time steps. For the problem shown in figure 1, the time savings was approximately 45%.
- **Assembly planning.** Consider the problem of finding a sequence in which to disassemble the simple five-block structure shown in the upper left corner of figure 5 so that it does not collapse. At one point in the planning algorithm it may be necessary to analyze the stability of the structure $\{1, 3, 5\}$; at another point, it may be necessary to analyze the stability of the structure $\{1, 3, 4, 5\}$. We show that it is possible to reuse the results of the

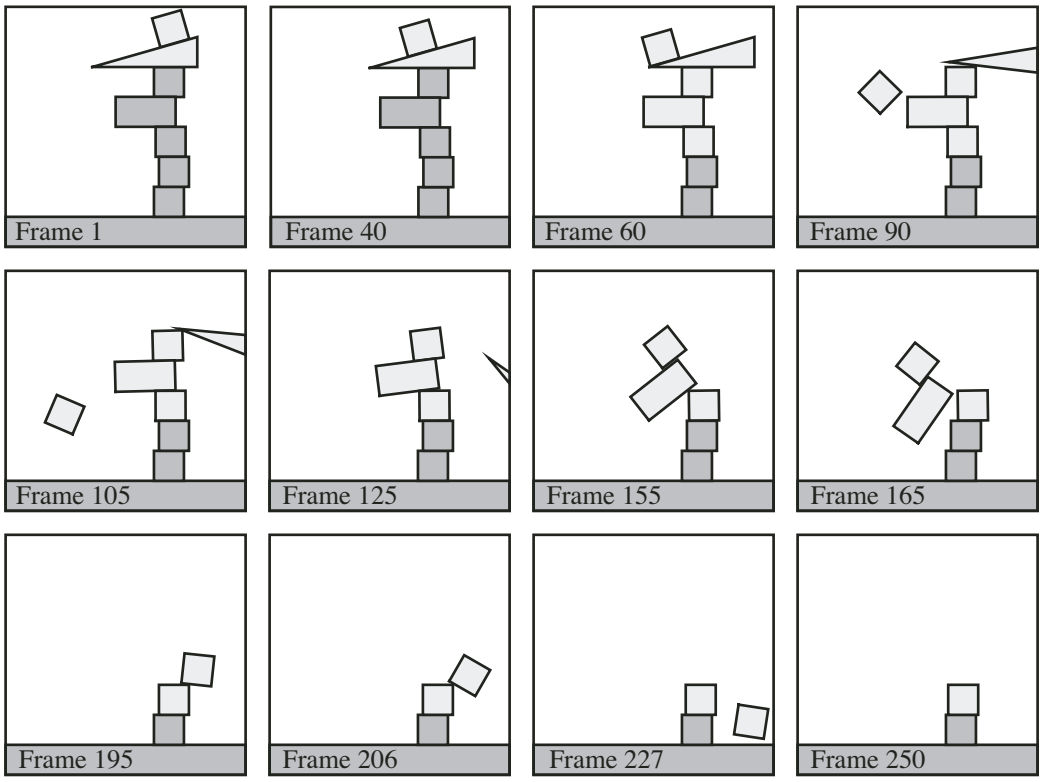

Fig. 1. A column falling apart. Dark gray blocks are motionless. The motionless set changes at frames 60 and 195.

stability computation for the smaller structure to quickly compute the stability of the larger structure.

A. Related Work

Formulation of rigid-body dynamics as a linear complementarity problem allows both principled analysis and robust simulation. Lötstedt was the first to show that if the system constraints are inequality constraints, *e.g.* contact forces, then the system dynamics can be modeled as a quadratic program [6] or a linear complementarity problem [1]. Recent formulations allow for impact and make additional guarantees about convergence; see Anitescu and Potra [7], and Stewart and Trinkle [3].

Dynamics problems for rigid bodies with a Coulomb friction model are complicated by the fact that, for particular configurations, the dynamics equations may have multiple solutions (indeterminacy) or no solutions at all (inconsistency), (Painlevé, 1895 [8]). In the present work, we assume that solution existence and uniqueness have been analyzed by other means. The problem of finding sufficient conditions for stability with friction has been explored by Pang and Trinkle [9], [10], who use a complementarity formulation to show that if friction coefficients are sufficiently small, a solution to the dynamics problem exists and is unique. In some cases, manipulation strategies may be chosen to avoid problems of frictional indeterminacy; see Balkcom and Trinkle [11], as well as Erdmann's [12] seminal work on force closure and friction grasps.

We explore two applications of computation reuse: dynamic simulation and assembly planning. Our algorithm for dynamic simulation is based on Baraff's algorithm [2] for computing contact forces in an assembly of rigid bodies; in the frictionless case, Baraff's algorithm reduces to Dantzig's algorithm. Within assembly-planning, subassembly stability has been studied by Boneschanscher *et al.* [13]. Wilson and Rit [14] consider computation reuse in the context of assembly problems, but focus on geometric computations to determine free motions of the bodies, and note that they "have not found such regularities for stability information."

II. The Principle Pivoting Method

In this section we describe Dantzig's algorithm, sometimes called the *principle pivoting method*. We first introduce some terminology commonly used when discussing LCPs. A solution $(\mathbf{f}, \mathbf{a})$ for equality 1 is *feasible* if it satisfies equation 2. A *complementary solution* is one that satisfies equation 3. Every solution technique for a linear complementarity problem seeks a *complementary feasible solution* for equation 1.

The principle pivoting method described by Cottle and Dantzig [15] begins by initializing $(\mathbf{f}, \mathbf{a})$ with the complementary solution $(\mathbf{0}, \mathbf{b})$. It progresses by means of *major* and *minor cycles*. If the problem is nontrivial, there exists an index i such that $a_i < 0$. The goal of a major cycle is to achieve feasibility for the pair (f_i, a_i). Within a major cycle, some number of minor cycles occur to ensure that feasibility and complementary conditions are maintained for all previously

considered indices. A minor cycle consists of a single *pivot operation*. At the end of the major cycle, the pair is feasible and complementary. The algorithm continues until no negative variables remain in $\mathbf{a}$.

III. Computation reuse

To solve the LCP $(\mathbf{A}, \mathbf{b})$, the principle pivoting method may execute as many as n major cycles, as shown in [15]. We show in this section that if we initialize the algorithm with a solution to a subassembly LCP, we reduce the maximum number of major cycles.

Theorem 1: Let $(\mathbf{A}, \mathbf{b})$ be an LCP, let $\mathbf{A}_{11}$ be any $k \times k$ principle submatrix of $\mathbf{A}$, and let $\mathbf{b}_1$ be a k-dimensional subvector of $\mathbf{b}$ with the same rows removed. If $(\mathbf{x}, \mathbf{y})$ is a complementary feasible solution to the LCP $(\mathbf{A}_{11}, \mathbf{b}_1)$, then initializing Dantzig's algorithm with the complementary solution $([\ \mathbf{x}\ \mathbf{0}\ ]^T, [\ \mathbf{y}\ \mathbf{A}_{21}\mathbf{x} + \mathbf{b}_2\ ]^T)$ will allow it to find a complementary feasible solution for the LCP $(\mathbf{A}, \mathbf{b})$ in at most $n - k$ major cycles.

Proof: If $(\mathbf{x}, \mathbf{y})$ is a complementary feasible solution to the LCP $(\mathbf{A}_{11}, \mathbf{b}_1)$, then there exists $\mathbf{a}$ such that $([\ \mathbf{x}\ \mathbf{0}\ ]^T, \mathbf{a})$ is a complementary solution to $(\mathbf{A}, \mathbf{b})$. Since contact ordering is arbitrary, we can arrange $\mathbf{A}$ and $\mathbf{b}$ such that $\mathbf{A}_{11}$ is the upper left $k \times k$ principle submatrix of $\mathbf{A}$, and $\mathbf{b}_1$ is the upper k-dimensional subvector of $\mathbf{b}$. Thus, we may partition $\mathbf{A}$ and $\mathbf{b}$ as in the equation

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{y} \\ \mathbf{A}_{21}\mathbf{x} + \mathbf{b}_2 \end{bmatrix}. \quad (5)$$

Since $(\mathbf{x}, \mathbf{y})$ is complementary, $(\mathbf{f}, \mathbf{a})$, where $\mathbf{f} = [\ \mathbf{x}\ \mathbf{0}\ ]^T, \mathbf{a} = [\ \mathbf{y}\ \mathbf{A}_{21}\mathbf{x} + \mathbf{b}_2\ ]^T$, is a complementary solution for $(\mathbf{A}, \mathbf{b})$.

Solving this equation, each major cycle reduces the number of negative variables in $\mathbf{a}$ by at least one. Since $\mathbf{y}$ is feasible, there are at most $n - k$ negative elements in $\mathbf{a}$. Thus, the algorithm can make no more than $n - k$ major cycles. ■

A. Reuse for physical structures

In this section, we show that we can reuse computation in a dynamics formulation for physical structures. Specifically, we show that a subproblem corresponds to the dynamics of a substructure.

The series of steps in figure 2 illustrate how we use Dantzig's algorithm to solve the system dynamics incrementally. In the first frame, the contact forces on a single block are computed without considering the force contributions of the other blocks. In the next frame a block is added, and the forces at c_1 and c_2 are adjusted as contact forces on the second block are computed. In the last frame, a final block is added, and the contact forces beneath the first two blocks are adjusted.

In order to reuse computation as discussed in this example, a subproblem of the system dynamics must correspond to a substructure. Specifically, we show that for the LCP $(\mathbf{A}, \mathbf{b})$ from Baraff's model of the system dynamics, described in equation 4, $(\mathbf{A}_{11}, \mathbf{b}_1)$ is the system dynamics of a substructure.

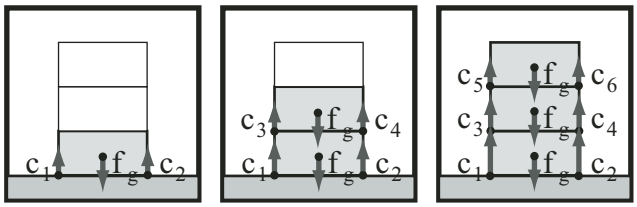

Fig. 2. Sequential computation of contact forces for subassemblies. c_i indicates the ith contact force and f_g the gravitational force. The length of an arrow shows the relative magnitude of the force it represents.

In equation 4, rows of $\mathbf{J}$ correspond to contacts in the structure and columns correspond to bodies, rows and columns of $\mathbf{M}$ correspond to bodies, as do rows of $\mathbf{F}^{\text{ext}}$. Thus, for a given structure and substructure, we can partition these terms as

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_{11} & \mathbf{J}_{12} \\ \mathbf{J}_{21} & \mathbf{J}_{22} \end{bmatrix}, \quad \mathbf{M}^{-1} = \begin{bmatrix} \mathbf{M}_{11}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_{22}^{-1} \end{bmatrix}, \quad \mathbf{F}^{\text{ext}} = \begin{bmatrix} \mathbf{F}_1^{\text{ext}} \\ \mathbf{F}_2^{\text{ext}} \end{bmatrix}, \quad (6)$$

where the top and left partitions correspond to elements present in the structure but not in the substructure. Then,

$$\begin{aligned} \mathbf{A}_{11} &= \mathbf{J}_{11}\mathbf{M}_{11}^{-1}\mathbf{J}_{11}^T + \mathbf{J}_{12}\mathbf{M}_{22}^{-1}\mathbf{J}_{12}^T, \\ \mathbf{b}_1 &= \mathbf{J}_{11}\mathbf{M}_{11}^{-1}\mathbf{F}_1 + \mathbf{J}_{12}\mathbf{M}_{22}^{-1}\mathbf{F}_2. \end{aligned} \quad (7)$$

However, $\mathbf{J}_{12} = \mathbf{0}$, since no contacts present in the substructure are contacts between objects that are not in the substructure. Thus equation 7 reduces to

$$\begin{aligned} \mathbf{A}_{11} &= \mathbf{J}_{11}\mathbf{M}_{11}^{-1}\mathbf{J}_{11}^T, \\ \mathbf{b}_1 &= \mathbf{J}_{11}\mathbf{M}_{11}^{-1}\mathbf{F}_1, \end{aligned} \quad (8)$$

the exact formulation of the substructure LCP.

IV. Application: dynamic simulation

During simulation, it is often the case that a subset of contacting rigid bodies remains motionless for many time steps. Reusing the persistent substructure's solution allows us to optimize the contact force calculation for the simulation as described in the previous section. Figure 3 illustrates this approach. We compute the dynamics to identify the set of motionless bodies, re-compute the dynamics of that subset as a partial solution, and use this partial solution as a starting point for computing the dynamics of the complete structure in successive time steps. If the set of motionless bodies changes, we compute the partial solution corresponding to the new motionless set, and use it until the motionless set changes again. This algorithm may be written

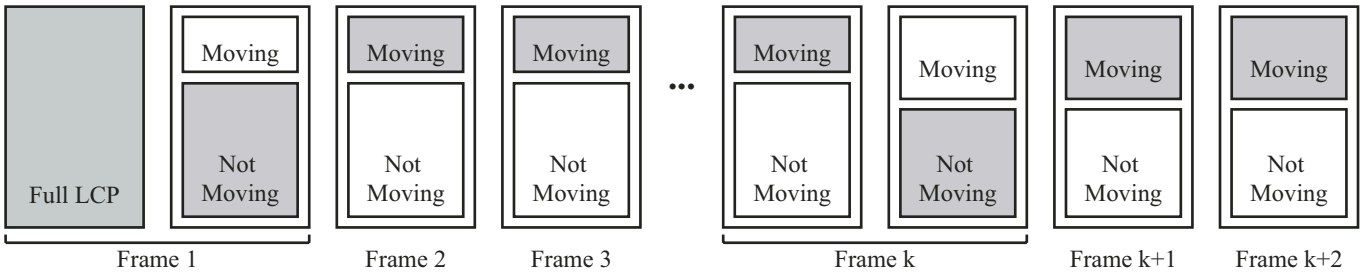

Fig. 3. Illustration of how an LCP solution may be reused in dynamic simulation. Shaded areas represent the portion of the LCP that must be evaluated at each step. In frame k, an element that was previously motionless begins to move, so we must solve the LCP corresponding to the new motionless set and store it for reuse in subsequent frames.

```
SIMULATE(world)
1  prevSoln ← ∅
2  prevMotionless ← ∅
3  time ← 0
4  while (time++ < SIMULATIONTIME)
5        do SOLVELCP(world, prevSoln)
6           motionless ← GETMOTIONLESSBODIES(world)
7           if (prevMotionless ≠ motionless)
8             then prevSoln ← SOLVELCP(motionless, ∅)
9                  prevMotionless ← motionless
```

At each time step, we apply the external and contact forces to find the acceleration of each body, and then integrate accelerations to find the location and velocity of the bodies in the next time step.

We have implemented this algorithm for a planar simulator with and without friction. The frictionless case reduces to Dantzig's algorithm. In the frictional case, we use Baraff's modification to Dantzig's algorithm [2]. Baraff's formulation is not a true LCP, as it also includes a number of auxiliary conditions in addition to the standard complementary and feasibility conditions of an LCP. This algorithm is not proven to converge, but is considered to be reliable in practice. The SIMULATE function with computation reuse is also applicable to Baraff's three-dimensional model, and to three-dimensional models that employ linearized friction cones, where $\mathbf{A}$ is a P-matrix or is PSD.

For the example in figure 1, simulated with a small coefficient of friction, our optimized algorithm reduced the time spent solving LCPs from 2.2 s to 1.2 s, a reduction of 45%. For those frames where five of the seven blocks are motionless, the average savings from computation reuse was 65%.

We expect the simulation reuse algorithm to be fastest when the motionless structure is large and when we are able to reuse a solution across many frames, as is the case when we use a small integration step. Thus, this algorithm allows a more fine-grained time step, and we expect it to be particularly useful for applications where a highly detailed simulation is required, such as mechanism analysis.

Although reducing the number of major cycles in Dantzig's algorithm usually improves the overall running time of the algorithm, this is not always the case. The number of minor cycles within a major cycle may depend on the order in which contacts are considered. We have observed that in some cases, the contact ordering imposed by the computation reuse algorithm causes major cycles to take more time. In the example in figure 4, there are three simulation segments. For the first 41 frames, only blocks a and b are in motion. In the next 46 frames, c, d, and e are in motion as well. For the remainder of the simulation, all blocks except for ground are moving. LCPs in the second segment take nearly four times as long to compute when we reuse computation, in spite of the fact that the algorithm makes only 22 major cycles instead of 60. This is due to a greater number of minor cycles in each major cycle.

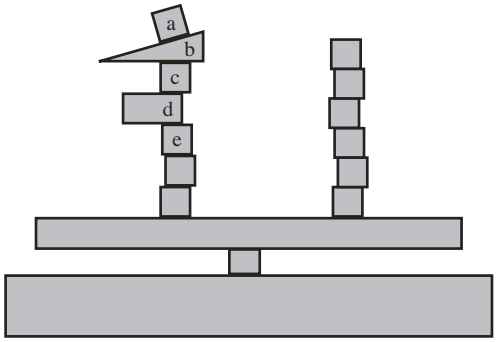

Fig. 4. Three different motionless sets appear in this simulation: first only a and b are in motion, then c, d, and e move as well, then all blocks fall.

V. APPLICATION: ASSEMBLY PLANNING

A fundamental problem in mechanical assembly is to find an order in which a product can be assembled or disassembled. In this section, we consider the problem of finding a *stable disassembly sequence*, that is, an order in which we can remove every object from the structure without causing it to collapse under gravity after any step. As in dynamic simulation, structures with substructures in identical configurations arise naturally in this problem. Researchers have considered reusing similar geometries for movability [14], [16], but, to our knowledge, not for determining stability. In automated assembly sequencing, it is typical to determine stability by solving a linear program (LP) corresponding to the static force-balance equations [17]. We show that our LCP stability

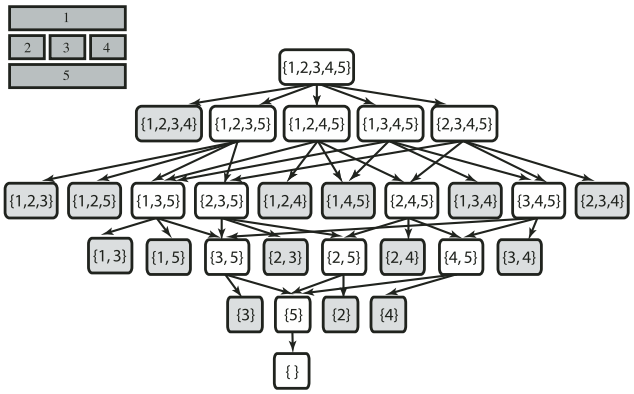

Fig. 5. A simple structure and its disassembly graph.

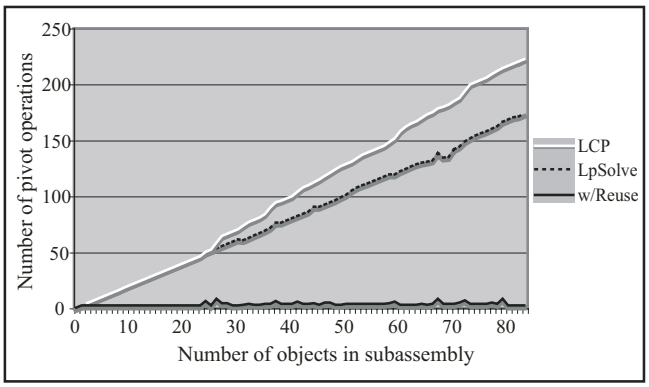

Fig. 6. Number of pivots required by a single stability test, by the size of the subassembly tested, for the 84-block randomly generated structure shown in figure 7.

test with computation reuse outperforms this approach, even though we expect system dynamics to be more complicated than statics, as illustrated by the difference between LpSolve and Dantzig LCP columns in table I.

A basic approach for finding a stable disassembly sequence is as follows. Consider the set of all objects in a structure. Collect all possible subsets into a *disassembly graph*, as shown in figure 5. Search the graph, testing the stability of each node as it is visited. Any path of stable nodes from start to goal is a stable disassembly sequence.

Many researchers have considered the problem of improving the performance of this algorithm by reducing the number of nodes visited [18], [14]; we focus on reducing the amount of time spent testing the stability of a node. We can test stability by computing a solution for the contact forces, and then verifying that body accelerations are zero. (If there is friction, this test is only a necessary condition for stability; computing *strong* stability is beyond the scope of this paper.)

Since we add a single block at a time, and only consider successors to stable nodes, we can use the complementary feasible solution from any parent as a starting point for computing the stability of the current node, reducing the number of major cycles from n or fewer to a small constant.

We have implemented an unstacking planner for planar systems with static friction. A depth-first search strategy was used to explore the assembly graph. The planner was configured to use one of three stability tests: solving the system statics, solving the system dynamics using Dantzig's algorithm, or solving the system dynamics using Dantzig's algorithm with our modifications for computation reuse. The system statics stability test was implemented with `lp_solve` – an open-source linear program solver that uses the revised simplex method [19] – to determine the feasibility of a structure's LP. Since `lp_solve` is written in C and our Dantzig implementation in Java, we used the number of pivot operations each algorithm made as our comparison metric. The time it takes to complete a pivot operation grows with the size of the problem, so our LCP algorithm with computation reuse is not truly a linear-time algorithm, but it did consistently outperform the other approaches, as shown in figure 6.

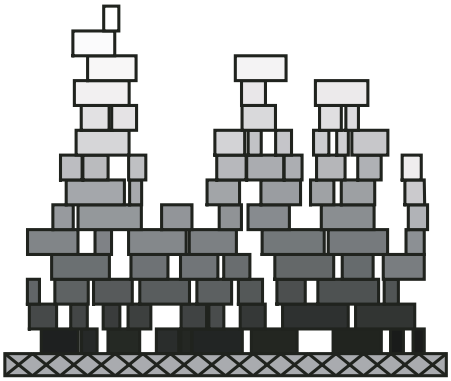

Fig. 7. A randomly generated assembly, where disassembly order is indicated by block color. Observe that blocks in the column on the far right must be removed before other blocks at higher levels in the structure.

TABLE I

TOTAL NUMBER OF PIVOTS BY ALGORITHM, FOR FRICTIONLESS EXAMPLES.

Structure	Blocks	LpSolve	Dantzig LCP	LCP + Reuse
Random	10	102	127	13
Column	50	2352	2352	96
Column	80	6162	6162	156
Random	84	11,975	15,136	420
Pyramid	91	8190	8190	180

We tested the different algorithms on a variety of examples, including columns, pyramids, and randomly generated structures, such as the one shown in figure 7. More detailed discussion of these examples can be found in [20]

Table I presents the number of pivots the algorithm made for a few example problems.

VI. FUTURE WORK

Robotics algorithms typically solve a sequence of dynamics problems to simulate, reason about, or control the behavior of a system of rigid bodies. This paper presented initial work on optimizing dynamics computations by exploiting structure in linear complementarity problems imposed by similarities

between physical systems.

Although our initial results are promising, there are a number of limitations to the approach, and many directions in which the work might be extended.

First, we have only considered Dantzig's algorithm. There are many other techniques for solving linear (and non-linear) complementarity problems, including Lemke's algorithm [4] and the PATH solver [5]. Dantzig's algorithm is guaranteed to find a solution if one exists, and $\mathbf{A}$ is either a P-matrix (has positive principle minors) or is positive semi-definite. Lemke's algorithm is applicable to a somewhat larger class of problems, and guaranteed to find a solution if $\mathbf{A}$ is a P-matrix or copositive-plus. In practice, Dantzig and Lemke often converge even in the case where $\mathbf{A}$ is not of a suitable class. It is not immediately clear how to reuse computation in Lemke's algorithm, unfortunately, as the algorithm is less incremental in nature than Dantzig's.

Second, although in the worst-case analysis, computation reuse is faster, there is no guarantee that it will be faster in any particular situation. The time required to execute each major cycle of Dantzig's algorithm is dependent on the order in which contacts are considered, and using the solution to a sub-LCP as input to the LCP arbitrarily fixes the order in which the first k contacts have been considered.

The assembly problem we considered is actually a statics problem, a special case of a dynamics problem. The problem of determining whether a motionless structure may remain motionless can be phrased as a linear program (LP) rather than as an LCP. The LP is simpler than the LCP, the LP matrix is typically sparse, and there exist robust software packages to solve very large LPs. Can computation be reused in LP solution methods? There is a well-known correspondence between LPs and LCPs; one approach would be to write the LP as an equivalent LCP, and use Dantzig's algorithm LCP to solve. Dantzig's [21] simplex algorithm is a more usual approach to solving LPs, but as with Lemke, it is not immediately clear how to reuse computation.

The examples we have considered hint at the breadth of problems for which computation reuse of the type we describe is a useful optimization. In future work, we plan to implement larger examples, and implement the dynamics simulation algorithm as a component of a complete rigid-body simulation package such as ODE [22].

ACKNOWLEDGMENT

The authors would like to thank Christopher Bailey-Kellogg, Amit Chakrabarti, Peng Song, Jeffrey Trinkle, and the members of the Dartmouth Robotics Laboratory for their guidance and insightful comments.

This research program is a part of the Institute for Security Technology Studies, supported by Grant No. 2005-DD-BX-1091 awarded by the Bureau of Justice Assistance. The Bureau of Justice Assistance is a component of the Office of Justice Programs, which also includes the Bureau of Justice Statistics, the National Institute of Justice, the Office of Juvenile Justice and Delinquency Prevention, and the Office for Victims of Crime. Points of view or opinions in this document are those of the author and do not represent the official position or policies of the United States Department of Justice.

REFERENCES

[1] P. Lötstedt, "Coulomb friction in two-dimensional rigid-body systems," *Zeitschrift für Angewandte Mathematik und Mechanik*, vol. 61, pp. 605–615, 1981.

[2] D. Baraff, "Fast contact force computation for nonpenetrating rigid bodies," in *SIGGRAPH*, 1994, pp. 23–34.

[3] D. Stewart and J. Trinkle, "An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and Coulomb friction," *International Journal of Numerical Methods in Engineering*, vol. 39, pp. 2673–2691, 1996.

[4] C. Lemke, "Bimatrix equilibrium points and mathematical programming," *Management Science*, vol. 11, no. 7, May 1965.

[5] S. Dirkse and M. Ferris, "The PATH solver: a non-monotone stabilization scheme for mixed complementarity problems," *Optimization Methods and Software*, vol. 5, pp. 123–156, 1995.

[6] P. Lötstedt, "Numerical simulation of time-dependent contact friction problems in rigid body mechanics," *SIAM Journal of Scientific Statistical Computing*, vol. 5, no. 2, pp. 370–393, 1984.

[7] M. Anitescu and F. Potra, "Formulating multi-rigid-body contact problems with friction as solvable linear complementarity problems," *ASME Journal of Nonlinear Dynamics*, vol. 14, pp. 231–247, 1997.

[8] P. Painlevé, "Sur les lois du frottemont de glissement," *C. R. Acad. Sci.*, vol. 121, pp. 112–115, 1895.

[9] J.-S. Pang and J. Trinkle, "Stability characterizations of rigid body contact problems with Coulomb friction," *Zeitschrift für Angewandte Mathematik und Mechanik*, vol. 80, no. 10, pp. 643–663, 2000.

[10] J. Trinkle, J. Tzitzouris, and J. Pang, "Dynamic multi-rigid-body systems with concurrent distributed contacts: theory and examples," *Philosophical transactions on mathematical, physical, and engineering sciences*, vol. 359, no. 1789, pp. 2575–2593, 2001.

[11] D. J. Balkcom and J. C. Trinkle, "Computing wrench cones for planar contact tasks," *International Journal of Robotics Research*, pp. 1053–1066, 2002.

[12] M. A. Erdmann and M. T. Mason, "An exploration of sensorless manipulation," *IEEE Transactions on Robotics and Automation*, vol. 4, no. 4, pp. 369–379, Aug. 1988.

[13] N. Bonschanscher, H. van der Drift, S. J. Buckley, and R. H. Taylor, "Subassembly stability," in *AAAI 88: The seventh national conference on Artificial Intelligence*, vol. 2, Aug. 1988, pp. 780–785.

[14] R. H. Wilson and J.-F. Rit, "Maintaining geometric dependencies in an assembly planner," in *IEEE International Conference on Robotics and Automation*, May 1990.

[15] R. W. Cottle and G. B. Dantzig, "Complementary pivot theory of mathematical programming," *Linear algebra and its applications*, vol. 1, pp. 103–125, 1968.

[16] R. Hoffman, "Automated assembly planning for B-rep products," in *Systems Engineering, 1990., IEEE International Conference on*, Pittburgh, PA, USA, Aug. 1990, pp. 391–394.

[17] B. Romney, "Atlas: an automatic assembly sequencing and fixturing system," in *Theory and Practice of Geometric Modeling*, 1997, pp. 397–415.

[18] C. J. M. Heemskerk, "The use of heuristics in assembly sequence planning," *Annals of the CIRP*, vol. 38, no. 1, pp. 37–40, 1989.

[19] M. Berkelaar, K. Eikland, and P. Notebaert, "lp_solve, open source (mixed-integer) linear programming system," May 2004, version 5.1.0.0.

[20] A. Loomis, "Computation reuse in statics and dynamics problems for assemblies of rigid bodies," Dartmouth College, Computer Science, Hanover, NH, Tech. Rep. TR2006-576, June 2006. [Online]. Available: ftp://ftp.cs.dartmouth.edu/TR/TR2005-576.pdf

[21] G. B. Dantzig, *Linear Programming and Extensions*. Princeton University Press, 1963.

[22] R. Smith, "Open dynamics engine: v0.5 user guide," May 2004.