MITSUBISHI ELECTRIC RESEARCH LABORATORIES
CAMBRIDGE RESEARCH CENTER

N-Body Spacetime Constraints

Diane Tang
Harvard University

J. Thomas Ngo
Interval Research Corp.

Joe Marks*
Mitsubishi Electric Research Laboratories, Inc.

TR-94-15 December 1994

Abstract

Animators frequently choreograph complex motions for multiple objects that interact through collision and obstruction. In such situations, the use of physically based dynamics to confer visual realism creates challenging computational problems. Typically forward simulation is well understood, but the inverse problem of motion synthesis—that of synthesizing motions consistent both with physical law and with the animator's requirements—is generally tedious and sometimes intractable. We show how N-body inverse problems can be formulated as optimization tasks. We present a simply stated, but combinatorially formidable example that exhibits all of the essential sources of complexity common to N-body motion synthesis, and show how it can be solved approximately using heuristic methods based on evolutionary computation.

Key Words: Animation, motion synthesis, heuristic methods, stochastic optimization, evolutionary computation, billiard-ball problems.

To appear in the Journal of Visualization and Computer Animation.

---

*The work described in this paper was done while JM was at Digital Equipment Corporation's Cambridge Research Lab.

1. First printing, TR94-15, July 1994.
2. Second printing, TR94-15a, December 1994.

Introduction

Simulating the dynamics of multibody physical systems is an established and challenging research topic in computer graphics [1, 2, 3, 10, 11, 18]. However, even if all the problems of physical modeling, collision detection, and efficient computation were to be solved satisfactorily, the resulting algorithms would still leave tedious tasks in the hands of human animators. The best tools for computer animation may not be those that just perform forward simulation well, but those that can also solve the inverse problem of *motion synthesis*: automatically finding trajectories for animated objects that both achieve the animator's goals and satisfy physical law (thus yielding visual plausibility).

Automatic motion synthesis for character animation has received considerable attention recently [5, 19, 20, 25, 26], but the notion of automatic motion synthesis for multibody systems, in which nonautonomous objects interact through collision and obstruction, has been largely ignored. In this paper we extend the Spacetime Constraints paradigm [26], originally developed for articulated-figure animation, to include multibody motion synthesis. The resulting class of problems is very broad. We present one simply stated problem that exhibits all the essential characteristics of this class, and describe an effective technique for solving it using heuristics based on evolutionary computation. We believe that this work can serve as a starting point for solving more interesting multibody motion-synthesis problems involving rigid and deformable 3D objects.

Problem Statement

Consider the following motion-synthesis problem, stated in terms of a set of N-body spacetime constraints. N disks (think of them as air-hockey pucks) are free to move without friction within a rectangular region of the plane. All collisions are elastic and frictionless. Gravity is absent. Given initial ($t = 0$) positions for the disks, the goal is to find initial velocities such that the disks arrive as close as possible[1] to the desired final ($t = t_{\text{final}}$)

[1]Whereas the original Witkin-Kass Spacetime Constraints formulation calls for the solution of an optimization problem with two explicit boundary conditions (the initial and desired final configurations), our approach uses only the initial configuration as a boundary condition; deviation from the desired final configuration is penalized by a term

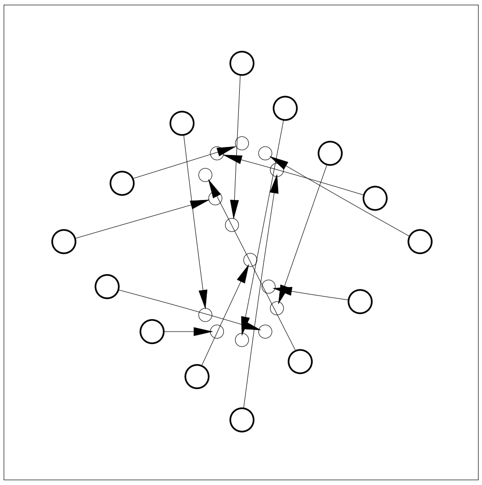

Figure 1: A sample 14-disk problem.

positions. (We employ an L^2, or sum-of-squares cost function to determine solution quality.) A sample problem is illustrated in Figure 1. In their initial positions, the disks describe a diamond shape; an arrow connects each disk to its target location, indicated by a small circle. Collectively, the target locations describe an 'S' shape.

As it has been stated thus far, the problem is neither overconstrained nor underconstrained; there are $2N$ control variables and $2N$ objective variables. The problem becomes overconstrained if some disks are declared to be *noncueable*, *i.e.*, unable to be given nonzero initial velocities. Note that a noncueable disk can begin to move only if hit by another disk. An overconstrained problem is shown in Figure 2, in which broken arrows connect initial and target positions of noncueable disks.

This simple N-disk problem exhibits all of the essential sources of complexity common to N-body spacetime-constrained problems:

- For visual realism, the objects move according to predetermined rules of motion (here, physical law).[2]

in the objective function. For this reason, the formal statement of our problem differs from that of Witkin and Kass—nonetheless, our overall goals are essentially identical.

[2]An alternative approach to motion synthesis for multiple moving bodies, exemplified

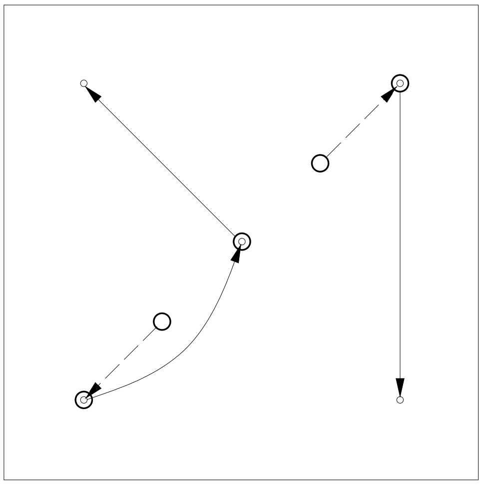

Figure 2: An overconstrained 5-disk problem.

- Control is limited. (In this case the locus of control is instantaneous, at time $t = 0$. Moreover, in the overconstrained version of the problem, only some disks are controllable.)
- The objects can interact (here, collide) with each other, precluding any general attempt to solve for their motions independently.

Analyses of similar problems [4, 14, 15, 21] have shown that simple physical systems like those considered here can exhibit chaotic behavior, and that the corresponding optimization problems can have daunting computational complexity—they may not even be computable! Nevertheless, general intractability results do not imply the difficulty of every instance in a class of problems: we have developed an algorithm that is capable of approximately solving nontrivial instances of the N-disk motion-synthesis problem. This algorithm is described in the next section.

by the DANCER system of Lomas and Todd [16], is to relax the requirement of physically correct dynamics by blending in the influence of inverse kinematics.

Algorithm

Motivating considerations

Our strategy in designing the overall architecture of our algorithm was to exploit properties of N-disk motion-synthesis that are likely to be common to other classes of N-body motion synthesis.

One such property is that the search problem, though cast in terms of continuous variables, is of a discrete character. That is, significantly improving a candidate solution often requires changing the pattern of disk-disk and disk-wall collisions, and local gradient information is of little use in choosing from among the possible patterns. This consideration led us to employ stochastic search, which does not rely on gradient information.

Another feature is that the problem has a limiting case in which it is completely decomposable: when the disk radii approach zero, the probability of disk-disk collision becomes negligible and each ball's motion can be synthesized independently. Moreover, this decomposability diminishes gradually as the disk radii are increased. Among stochastic search techniques, the genetic algorithm (GA) is best suited for such nearly decomposable problems [12]. Our algorithm is based on a standard form of GA, the steady-state GA [7, 9].

A third feature—the feature that our customization of the GA is tailored specifically to exploit—is that the form of the decomposition is mostly known in advance. When a problem is said to be decomposable, what is generally meant is that the objective function is approximately equal to a sum of terms, each of which depends on a small subset of the independent variables in a candidate solution. According to dogma [12], a GA implicitly discovers the mapping between terms and variables. With N-disk motion synthesis, however, much of the computational effort spent discovering this mapping would be wasted, since it is known in advance that the final position of a disk, and therefore the contribution of that disk to the L^2 error function, is determined largely by the initial velocity imparted to that disk. We exploit this known information about the problem's structure by introducing the concept of a trial solution's *signature*, described below.

Related work in multiobjective optimization

N-body motion synthesis is an example of *multiobjective optimization*: several goals exist, and success in meeting a subset of the goals does not obviate the need to meet the remaining goals. In such cases, it is often insufficient merely to combine the goals into a single objective function because the optimum of the function might be a solution in which only a subset of the goals are satisfied. Because of its potential for maintaining multiple candidate solutions in parallel, the GA has been explored as a basis for solving this general class of problems.

An early practical attempt at solving these problems in a general manner, called the Vector-Evaluated GA (VEGA) [23], differed from a standard GA in that each generation of individuals was constructed by pooling equal-sized subpopulations, each formed by selecting individuals according to a different one of the objective functions. It was later shown [22] that this strategy is equivalent to combining the multiple objectives linearly, albeit with weighting coefficients that depend on the contents of the population.

A more refined strategy for multiobjective optimization was proposed by Goldberg [9] and later implemented independently by Fonseca and Fleming [8] and by Horn et al. [13]. It involved the concept of *domination*: one solution is said to be dominated by another if the latter is better than the former in terms of all objectives. Nondomination is closely related to our concept of *subsumption*: one solution is said to be subsumed by another if the set of goals met by the latter is a superset of the goals met by the former. The strategy proposed by Goldberg also employed the biologically inspired concept of *niching,* which is related to our use of "Senate-like" representation, defined below. In niching, the objective function used to evaluate a given solution is influenced by the presence of similar solutions in the population. The biological metaphor is that similar solutions compete for like resources, which prevents any one species from filling the population.

Compared to our algorithm, none of the existing approaches completely exploits the third feature of N-body problems described above, namely that an approximate 1:1 mapping between the independent variables of the search and the multiple goals present in the objective function is known in advance.

Form of the algorithm

The pseudocode below is sufficiently generic to describe both our algorithm and the standard steady-state GA:

Initialize population of trial solutions
do
  Generate a new trial solution
  Insert it into the population, possibly
    deleting a current member of the population
while the best solution encountered is not bettered in K iterations

Despite superficial similarities, our algorithm cannot be considered a GA because it accomplishes in a greedy manner much of what is done in random fashion by a GA. In particular, our algorithm does not rely for its success on the validity of the Schema Theorem (the "fundamental theorem" for GAs) [12]. Many of the evolutionary phenomena that commonly cause a GA to converge on highly suboptimal solutions are absent in our algorithm.

The differences between our algorithm and a GA revolve around the use of signature information. In the following subsections we define signature information, then describe in detail how it is used in each of the italicized operations (*initialize*, *generate*, and *insert*).

Signature information

The signature of a trial solution is a bit vector in which each bit represents whether or not one of the objects in the multibody motion behaves in accordance with the animator's goals, within some predefined margin of error. In the context of the N-disk problem, let the *target neighborhood* of a disk be a circle of radius R centered on the disk's target location.[3] A trial solution is said to be *i-acceptable* if disk i is in its target neighborhood at time $t = t_{\text{final}}$. Bit i in a solution's signature is set if and only if the solution is i-acceptable. Also, the *order* of a solution is the number of bits set in its signature. As we shall see below, both signature and order are used to index trial solutions.

[3]In the runs presented here, R was set arbitrarily to 75 distance units. The disks themselves have radius $r = 20$ in the 14-disk problem and $r = 10$ in the 5-disk problem, and the bounding square is 800×800.

Our heuristics are based on the assumption that for any cueable disk i, the i-acceptability of a solution depends solely on the initial velocity of disk i. This assumption of independence is correct in the limiting case in which the disks have infinitesimal radius and therefore do not interact, but deteriorates as disk radius increases. For a problem with N infinitesimal disks, all of which are cueable, our algorithm is expected to converge very rapidly to a perfect solution.[4] When these conditions are relaxed, the performance of our algorithm is expected to degrade gracefully.

The *initialize* operation

A candidate solution is represented by an array of initial velocities, one for each cueable disk. Two methods for setting the initial velocity of a cueable disk are common to the *initialize* and *generate* procedures. *Directed randomize* chooses an initial velocity such that the cueable disk, were it to travel without colliding with any other disk, would arrive precisely at its target location at $t = t_{\text{final}}$ without colliding with any walls.[5] *Uniform randomize* chooses an initial velocity such that the cueable disk, were it to travel without colliding with any other disk, would have a uniform probability of arriving at any point within the rectangular boundary, possibly after bouncing off up to five walls.

The *initialize* procedure is designed to generate a diverse initial population with a roughly equal number of i-acceptable solutions for each i. Random trial solutions are added to the population, which is initially empty. Each random trial solution S is generated as follows: for each cueable disk, an initial velocity is chosen either by *directed randomize* or by *uniform randomize*, each with equal probability. S is rejected if its order is zero; otherwise, it is inserted. If, as a result, the population's membership or the number of solutions with the same signature as S exceeds a predefined limit (200 and 10, respectively, in our runs), the worst solution with the same signature as

[4] We tested this assertion on a variant of the problem in Figure 1 in which each disk has a very small radius. The algorithm found a perfect solution after just 54 iterations of the *generate-insert* loop.

[5] The notion of directed randomization could be generalized in several ways, (*e.g.*, by having cueable disks bounce off one or more walls on the way to their target locations, or by deliberately aiming cueable disks at noncueable ones in overconstrained problems like the one in Figure 2), but we have not yet explored generalizations of the basic idea.

S (possibly S itself) is deleted to make room.[6] The process continues until, for every i, the number of i-acceptable solutions in the population meets or exceeds a predefined limit (5, in our runs)—or, if this diversity criterion is never met, after a predetermined number of iterations.[7]

The *generate* operation

The *generate* procedure builds a new trial solution by executing one of the following four operations, each with equal probability: *creep*, *directed randomize*, *uniform randomize*, or *crossover*. The first three operations employ signature information solely to encourage even coverage of the search space. An existing population member (called the parent) is chosen such that every order o ($1 \leq o \leq N$) that is represented in the population has an equal probability of being chosen, and every signature of order o that is represented is as likely to be chosen as any other signature of order o. Thus, minority high-order signatures are given "Senate-like" representation during parent selection.[8] To generate the new trial solution (called the child), the initial velocity of one randomly selected cueable disk is perturbed slightly (in the case of *creep*), subjected to *directed randomize*, or subjected to *uniform randomize*.

The *crossover* operation, one of the hallmarks of a GA, requires the selection of two parents. The first is chosen as just described. Signatures are used in a slightly different manner to choose the second parent: a "Senate-like" distribution is again used, except that the only solutions with a chance of

[6]For this reason, the limit on the population's membership (200 in this case) should be set to a number greater than the product of the limit on the number of solutions with a given signature (10 in this case) and the number of bits in a signature (14 in the case of the 14-disk problem). Otherwise, there can be some i for which no i-acceptable solution is found before the population-membership limit is violated. Thereafter, all i-acceptable solutions created by the *initialize* operation will be rejected.

[7]In the two example problems illustrated in Figures 1 and 2, the diversity criterion was satisfied quickly in the initialization process, which produced average initial population sizes of 58.4 and 64.4, respectively.

[8]In the US Senate, each state is allocated two senators, regardless of population. In the US House of Representatives, the number of congressmen allocated to each state is proportional to population. With a "House-like" distribution, underrepresented solutions (for example, the only i-acceptable solution for a given i or the only nearly perfect solution) would rarely be selected as parents.

being selected are those with signature s_2 such that the number of bits set in $s_1|s_2$ (the "logical or" of s_1 and s_2) is the maximum possible. Signatures are used a third time in combining the two parents to "genetically engineer" a child: the initial velocity of cueable disk i is taken from the parent that is i-acceptable if exactly one such parent exists; otherwise, from either parent at random.

The *insert* operation

In the last procedure to consider, *insert*, signature information is used to avoid the deleterious proliferation of identical or nearly identical trial solutions, which is common in GAs. The signature of a child produced by *generate* is compared with the signature(s) of its parent(s). If the child's signature is the same as that of exactly one parent, the child replaces that parent, unless the parent is the best solution encountered so far and the child is not an improvement. (This is the strategy of *elitism* [7]: the best solution in the population is guaranteed not to be deleted.) If the child is a product of crossover and has the same signature as both parents, then the same procedure is used, except that one of the parents, selected at random, is considered for displacement.

If a parent is not displaced, either because of elitism or because the child's signature differs from its parent's (or parents'), the child is inserted into the population, replacing the worst solution with the same signature if the number of solutions with that signature exceeds a specified limit, just as in the *initialize* procedure.

If the number of solutions with the same signature is below the limit, then the insertion of the child can cause the overall population size to grow. Because the number of distinct signatures is 2^N, the potential for population growth is great, and this can be harmful: the bigger the population, the less time spent considering the most promising trial solutions. Some mechanism is therefore needed to limit this growth. Our solution is to delete every solution with the lowest-order signature that is subsumed by higher-order solutions, where we define subsumption to mean that every bit i that is set in the solution's signature is also set in some higher-order solution. (The only exception to this rule is that the all solutions with the same signature as the best solution in the population are exempt from deletion: this is another manifestation of elitism.) This culling strategy is triggered whenever the population size exceeds the predefined limit.

Results

The two test problems

For the problem depicted in Figure 1, our algorithm found initial velocities that lead to the disk trajectories shown in Figure 3. The disks are too large for all to move in straight lines from initial to target positions without colliding and thus disrupting the motion,[9] but the algorithm finds a collision-free motion after considering fewer than 10^6 trial solutions. Some disks bounce off three walls in the solution shown. To give some indication of how difficult this particular problem is, we can derive a crude lower bound on the size of the search space: a standard mathematical trick involving image targets reveals that in the absence of collisions between disks, the number of candidate solutions in which disks bounce against at most two walls is greater than 9^{14}, or 2×10^{13}.

Results for the overconstrained problem in Figure 2, in which disk-disk collisions must be exploited to achieve satisfactory results, are depicted in Figure 4. This solution is also the best found after consideration of fewer than 10^6 trial solutions.

The cost curves in Figures 5 and 6 illustrate the typical progress of the GA over time by showing the cost of the best solution encountered so far as a function of the number of trial solutions considered. An initial phase of rapid improvement is followed by long periods of very slow improvement that are punctuated by sharp, discrete decreases in cost. The changes in the nature and rate of progress over time suggest that the search algorithm might benefit from also being changed over time (*e.g.*, by increasing the use of *creep* as the *generate* operation in the latter half of the process), but this is an idea we have not yet pursued.

[9] Recall that no disk can be influenced after time $t = 0$. If the timing of the initial impulses could be varied, the two test problems considered here would be somewhat easier.

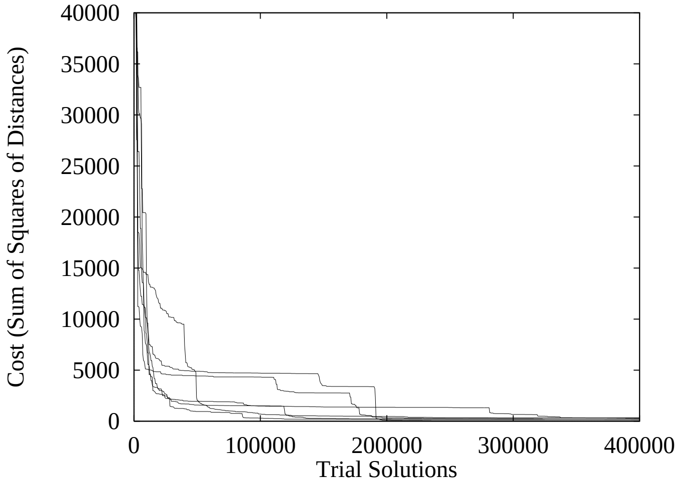

Figure 5: Six cost curves for the 14-disk problem.

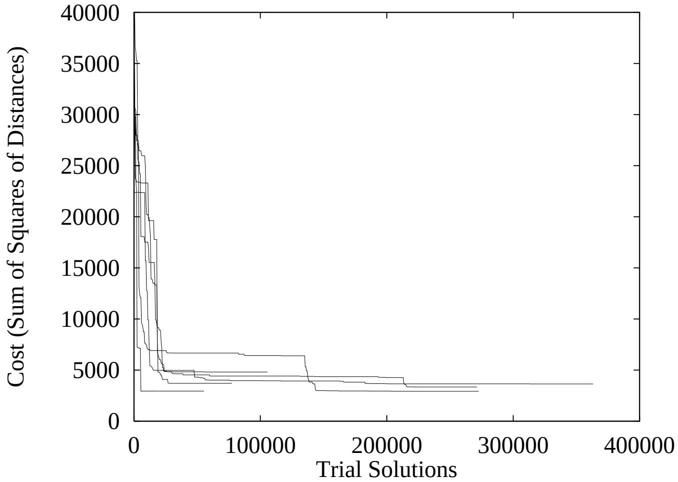

Figure 6: Six cost curves for the 5-disk problem.

Algorithm variants

Although the differences between our algorithm and a more standard GA center around a common simple theme, in terms of implementation our algorithm is significantly more elaborate than a GA. It is therefore reasonable to ask whether the added complication is of any significant practical value. To test the hypothesis that our algorithm is unnecessarily intricate, we concocted seven simpler variants, each of which disables one or more significant aspects of the original algorithm. The seven derivative algorithms result from the following simplifications:

- Uniform Probability for Parent Selection (B): In the *generate* operation, all trial solutions in the population are made equally likely to be selected as parents, instead of the signature-based selection scheme described above.

- No Crossover Operation (C): Crossover is removed as one of the possible *generate* operations, leaving only *creep*, *uniform randomize*, and *directed randomize*.

- No "Genetically Engineered" Crossover (D): Crossover is simplified by selecting velocities for the child from either parent randomly, instead of considering signature information to choose from the parent with the apparently better velocity value.

- No Signature Information (E): All aspects of the signature idea are removed, which can be achieved easily by setting the radius R of the target neighborhoods to 800, the width of the bounding square, in the original algorithm. (This variant has no intrinsic mechanism to limit the size of the initial population, so to facilitate comparisons with the original algorithm and the other variants, an initial population size of 60 was used.)

- No Culling of Subsumed Solutions (F): Population growth is curtailed in the *insert* operation by simply removing the worst trial solution in the population whenever a new solution is inserted.

- No Elitism (G): No provision is made to ensure that the best solution seen so far cannot be deleted from the population.

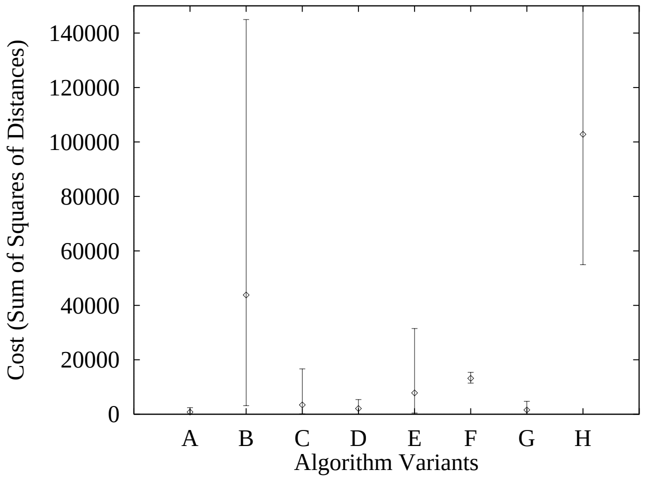

Figure 7: Comparative performance of various simplified algorithms for the 14-disk case. Algorithm A is the original algorithm; variants B–H are described in the text. Algorithm performance is stated in terms of the L^2 metric. The chart shows the average, best, and worst solutions found in five runs of 400,000 trial solutions each.

- No Directed Randomization (H): The *initialize* and *generate* operations cannot use *directed randomize*.

The issue of algorithm simplification is considered in more detail elsewhere [24], but Figures 7 and 8 show that none of these simplifications can be employed without compromising performance on at least one of the two problems considered here.

Conclusions

Extending the Spacetime Constraints paradigm of Witkin and Kass [26] to physical systems comprising multiple colliding bodies requires new inverse techniques to complement existing forward-simulation algorithms. We have considered one class of such problems, and demonstrated an effective heuristic technique, based on evolutionary computation, for this class. Our technique,

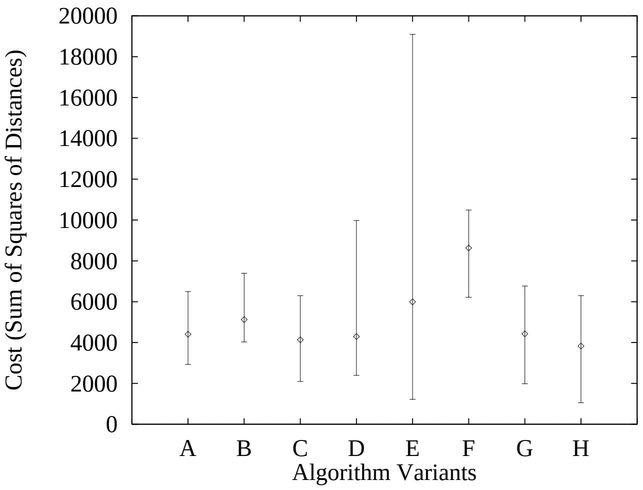

Figure 8: Comparative performance of various simplified algorithms for the 5-disk case.

which is closely related to the GA, differs from it in its use of signature information. In particular, our algorithm does not rely for its success on the so-called Schema Theorem [12], which is considered by many to be fundamental to the operation of a GA. Unlike a GA, our algorithm is expected to converge very rapidly to a perfect solution in the limiting case in which the moving bodies cannot interact. We speculate that variants of our algorithm may be applicable to other N-body problems in which (a) some degree of independence among the moving objects can be expected, and (b) methods for solving the corresponding one-body motion-synthesis problem exist.

As a strawman proposal, we propose two 3D N-body motion-synthesis problems as future challenges:

1. 10-pin bowling: The goal is to produce an animation in which a bowling ball knocks down a specified subset of the 10 pins arranged in the standard format.

2. Multivehicle car crash: Several moving vehicles participate in a crash in which the initial velocity and fate of each vehicle (*i.e.*, whether and how it tumbles, where it ends up, how many other vehicles it hits, etc.)

is determined by the animation script.

Aside from the consequences of requiring more complex physical simulation in 3D, these problems differ from those considered in this paper in two significant ways: the bowling problem has only one set of control variables, those relating to the bowling ball; and cars must be modeled as partially autonomous vehicles to achieve plausible effects. Both of these characteristics invalidate some of the assumptions we have made in developing our algorithm (*e.g.*, crossover cannot be applied when there is only one velocity vector—for the bowling ball—in a trial solution, and autonomic capabilities are not considered by the current algorithm). However, we believe that some of the ideas described here can be applied fruitfully to problems of this kind,[10] and we are optimistic that general N-body motion synthesis can be made practically useful in the future.

[10]Further supporting evidence for this belief is provided by recent work in assembly automation. "Shake-and-Make" assembly (more recently termed "mass aggregate assembly") [17] is a technique for assembling objects by agitating the parts: the parts are literally shaken together until assembly occurs! The corresponding optimization problem is to shape and position the parts so that the agitation will achieve the desired assembly most of the time. Chapman [6] has shown how a simple problem of this type, that of designing the shape of a feeder track, can be solved using a genetic algorithm.

Acknowledgments

DT is grateful for a Ford Grant that supported her initial work on this problem. We would like to thank Prof. Kung for the use of his computers to perform our experiments, and the members of the Harvard Animopt Group for suggestions and support.

References

[1] D. Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. *Computer Graphics*, 23(3):223–231, July 1989.

[2] D. Baraff. Coping with friction for non-penetrating rigid body simulation. *Computer Graphics*, 25(4):31–40, July 1991.

[3] D. Baraff. Issues in computing contact forces for non-penetrating rigid bodies. *Algorithmica*, 10(2/3/4):292–351, August/September/October 1993.

[4] C. Boldrighini, M. Keane, and F. Marchetti. Billiards in polygons. *The Annals of Probability*, 6(4):532–540, 1978.

[5] L. S. Brotman and A. N. Netravali. Motion interpolation by optimal control. *Computer Graphics*, 22(4):309–315, August 1988.

[6] C. D. Chapman. Genetic algorithm-based shape optimization for automated assembly. Technical Report GA-93-01, Massachusetts Institute of Technology, Department of Mechanical Engineering, Computer-Aided Design Lab, 1993.

[7] L. Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, NY, 1991.

[8] C. M. Fonseca and P. J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, 1993.

[9] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[10] J. K. Hahn. Realistic animation of rigid bodies. *Computer Graphics*, 22(4):299–308, August 1988.

[11] B. V. Herzen, A. H. Barr, and H. R. Zatz. Geometric collisions for time-dependent parametric surfaces. *Computer Graphics*, 24(4):39–48, August 1990.

[12] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975.

[13] J. Horn, N. Nafpliotis, and D. E. Goldberg. A niched pareto genetic algorithm for multiobjective optimization. *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Conference on Computational Intelligence*, 1:82–87, 1994.

[14] B. A. Huberman and P. Struss. Chaos, qualitative reasoning, and the predictability problem. In B. Faltings and P. Struss, editors, *Recent Advances in Qualitative Physics*, chapter 8, pages 119–135. MIT Press, Cambridge, MA, 1992.

[15] S. Kerckhoff, H. Masur, and J. Smillie. A rational billiard flow is uniquely ergodic in almost every direction. *Bulletin of the American Mathematical Society*, 13(2):141–142, October 1985.

[16] A. Lomas and S. Todd. DANCER: Behavioral goal directed motion control system for 3D computer animation. IBM United Kingdom Scientific Centre Report Number 271, September 1992.

[17] P. H. Moncevicz, M. J. Jakiela, and K. T. Ulrich. Orientation and insertion of randomly presented parts using vibratory agitation. In *Proceedings of the ASME Third Conference on Flexible Assembly Systems, DE-VOL 33*, pages 41–47, Miami, FL, September 1991. American Society of Mechanical Engineers.

[18] M. Moore and J. Wilhelms. Collision detection and response for computer animation. *Computer Graphics*, 22(4):289–298, August 1988.

[19] J. T. Ngo and J. Marks. Physically realistic motion synthesis in animation. *Evolutionary Computation*, 1(3):235–268, 1993.

[20] J. T. Ngo and J. Marks. Spacetime constraints revisited. In *SIGGRAPH '93 Conference Proceedings*, pages 343–350, Anaheim, CA, August 1993. ACM SIGGRAPH.

[21] J. H. Reif, J. D. Tygar, and A. Yoshida. The computability and complexity of optical beam tracing. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, pages 106–114. IEEE Computer Society, 1990.

[22] J. T. Richardson, M. R. Palmer, G. Liepins, and M. Hilliard. Some guidelines for genetic algorithms with penalty functions. *Proceedings of the Third International Conference on Genetic Algorithms*, pages 191–197, 1989.

[23] J. D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. *Proceedings of the [First] International Conference on Genetic Algorithms and Their Applications*, pages 93–100, 1985.

[24] D. Tang, J. T. Ngo, and J. Marks. Signature information and N-body spacetime constraints. Unpublished manuscript.

[25] M. van de Panne and E. Fiume. Sensor-actuator networks. In *SIGGRAPH '93 Conference Proceedings*, pages 335–342, Anaheim, CA, August 1993. ACM SIGGRAPH.

[26] A. Witkin and M. Kass. Spacetime constraints. *Computer Graphics*, 22(4):159–168, August 1988.