International Journal of Engineering, Science and Mathematics
Vol. 8 Issue 2, February 2019,
ISSN: 2320-0294 Impact Factor: 6.765
Journal Homepage: http://www.ijesm.co.in, Email: ijesmj@gmail.com
Double-Blind Peer Reviewed Refereed Open Access International Journal - Included in the International Serial Directories Indexed & Listed at: Ulrich's Periodicals Directory ©, U.S.A., Open J-Gage as well as in Cabell's Directories of Publishing Opportunities, U.S.A

LAZY ADDITION OF CONSTRAINTS

DR. VINOD KUMAR YADAV
RESEARCH SCHOLAR
J.P.U.CHAPRA
(DEPT.OF MATHEMATICS)

ABSTRACT

The preceding methods are all conservative in the sense that for each pair of objects there is always a linear constraint in the solver ensuring that the of objects will not overlap. We study the complex Combinatorial optimization problem to schedule the activities of a single project with the objective to complete the project with in the shortest -poss.

INTRODUCTION

The preceding methods are all conservative in the sense that for each pair of objects there is always a linear constraint in the solver ensuring that the objects will not overlap. However, if the objects are not near each other this incurs the overhead of keeping an inactive inequality in the linear solver. A potentially more efficient approach is to lazily add the linear constraints only if the objects are "sufficiently" close and remove them once they are sufficiently far apart.

We have investigated two variants of this idea which differ in the meaning of sufficiently close. The first method measures closeness by using the bounding boxes of the polygons. If these overlap, a linear approximation for the complex constraint is added and once they stop overlapping it is removed. We also investigated a more precise form of closeness, based on the intersection of the actual polygons, rather than the bounding box. However, we found the overhead involved in detecting intersection and the instability introduced by repeatedly adding and removing the non- overlap constraint made this approach infeasible. Thus we focus on the first variant.

Implementation relies on an efficient method for determining if the bounding boxes of the polygons overlap. Determining if n 2-D bodies overlap is a well studied problem and numerous algorithms and data structures devised including quad and dynamic versions of structures such as range, segment and interval-tress Unfortunately many of these methods handle non-rectangular shapes poorly, or are very difficult to implement. The method we have chosen to use is an adaptation of that presented in.

The algorithm is based, as with most efficient rectangle-intersection solutions, on the observation that two rectangles in some number of dimensions will intersect if and only if the span of the rectangles intersect in every dimension. Thus, maintaining a set of interacting rectangles is equivalent to maintaining two sets of intersecting intervals.

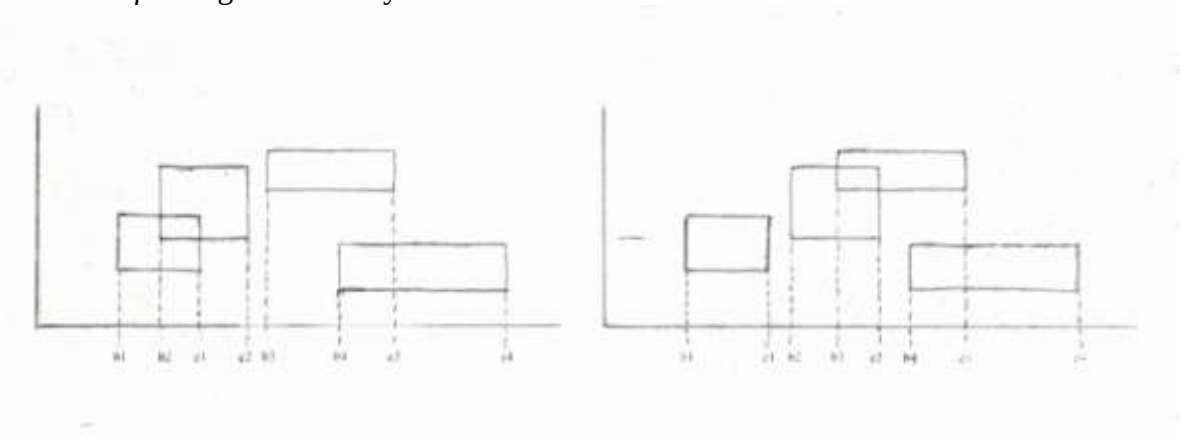
The algorithm acts by first building a sorted list of rectangle endpoints, and marking corresponding pairs to denote whether or not they are intersecting in either dimension. While this step takes, in the worst case $O(n^2)$ time for n rectangles, it is in general significantly faster. As shapes are moved, the list must be maintained in sorted order, and interacting pairs updated. This is done by using insertion sort at each time-step, which will sort an almost sorted list is O(n) time.

Note that it is undesirable to remove the liner constraint enforcing non-overlap between two polygons as soon as the solver moves them apart and their bounding boxes no longer intersect; instead such pairs of polygons are added to a removal buffer, and then removed

only if their bounding boxes are still not intersecting after the solver has reached a stable solution.

A change in intersection is registered only when a left and right endpoint of different bounding boxes swap positions, If a left endpoint is shifted to the left of a right endpoint, an intersection is added if and only if the boxes are already intersecting in all other If a left endpoint is shifted to the right of a right endpoint, the pair cannot intersect, so the pair is added to the removel buffer.

Unfortunately, we have found that this simple approach to lazy addition of constraints has the significant drawback of violating the conservativeness of the approximation and somewhat undermines the smoothness of the approximation since objects can momentarily overlap during direct manipulation. This can cause problems when the objects are being moved rapidly; that is, the distance moved between solves is large compared to the size of the objects. This is not very noticeable with the inverse approach but is quite noticeable with the decomposition method, as the convex components are often rather small; if two shapes are moved sufficiently far between solves, the local selection of configurations may be unsatisfiable**.**



The sorted list of endpoints is kept facilitate detection of changes in inter-section. As the second box moves right,b_2 moves to the right of e_1, which means that boxes 1 and 2 can no longer intersect. Conversely, endpoint $e_2$moves to the right of b_3 which means that boxes 2 and 3 may now intersect.

One possible solution would be to approximate the shapes by a larger rectangle with some padding" around each of the subjects. Another possible approaches to use rollback to recover from overlapping objects. When overlap is detected using collision detection, we roll back to the previous desired values. Add the non -overlap constraint and re-solve and, finally

solve for the current desired value. This should maintain most of the speed benefits of the current lazy addition approaches, while maintaining conservativeness of approximation; and using a separate layer for the late addition avoids adding additional complexity to the linear constraint solver.

REFERENCES

1. BADROS, G.J AND BORNING, A.2001. Cassowary constraint solving toolkit. Web page. http: //www. cs.washington.edu/research/constraints/cassowary.

2. BARAFF,D.1994. Fast contact force computation for nonpenetrating rigid bodies. In SIGGRAPH '94 Confer-ence Proceeding. ACM, New York, 23-32.
3. BORNING, A., MARRIOTT, K; STUCKEY, P; AND XIAO, Y.1997. Solving linear arthimetic constraints for user interface applications. In Proceedingof the 1997 ACM Symposium on User interface Software and Technology. ACM, New York.
4. MYERS, B.A 1996. The Amulet user interface development environment. In CHI'96 Conference Companion: Human Factors in computing System. ACM,New York.
5. SANNELLA, M.,MALONEY, J., FREEMAN-BENSON, B., AND BORNING , A. 1993. Multi-way versus one-way constraints in user interfaces: Experience with the DeltaBule aglorithm. Software- Practice and Experience 23,5(May),529-566.
6. BORNING, A, LIN,R, AND MARRIOTT, K, 1997**.** Constraints for the web. In proceedings of ACM MULTI-MEDIA'97.
7. *BORNING, A., LIN, R., AND MARRIOTT, K. 2016**.** Constraint-based document layout for the web.Multimedia Systems 8, 3, 177-189.*
8. *SANNELLA, M. 1994**.S**ky BLUE: A multi-waylocal propagation constraint solver for user interface construction. In proceedings of the 1994 ACM Symposium on User interface Software and Technology ACM, New York, 137-146.*
9. ***H**arvey, W., Stuckey, P., Borning, A.; Compiling constraint solving using projection in Principles and Practice of Constraint Programming –CP97, vol 1330 of LNCS, pp. 491-505. Springer, 1997.*