Graph Partitioning Algorithms for Rigid Body Simulations

Yinchu Liu and Sheldon Andrews

École de technologie supérieure, Montreal, Quebec, Canada

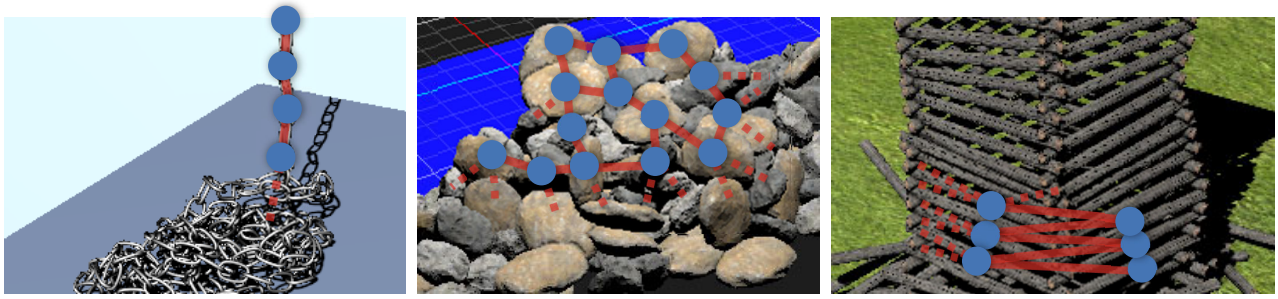

Figure 1: *Rigid body simulations have graph representations comprised of rigid body nodes (blue) connected by edges (red) due to a constraint topology. Our work proposes several graph partitioning algorithms to decompose the simulation into smaller sub-systems that may be solved in parallel. Here, graphs are hallucinated for several examples: Chain Drop (left), Rock Pile (middle), Log Tower (right).*

Abstract

We propose several graph partitioning algorithms for improving the performance of rigid body simulations. The algorithms operate on the graph formed by rigid bodies (nodes) and constraints (edges), producing non-overlapping and contiguous sub-systems that can be simulated in parallel by a domain decomposition technique. We demonstrate that certain partitioning algorithms reduce the computational time of the solver, and graph refinement techniques that reduce coupling between sub-systems, such as the Kernighan–Lin and Fiduccia–Mattheyses algorithms, give additional performance improvements.

CCS Concepts

• ***Computing methodologies*** → ***Physical simulation;*** • ***Mathematics of computing*** → *Graph algorithms;*

1 Introduction

Simulating rigid bodies with contact is a long-standing challenge in physics-based animation. A common approach is to model the system as a constrained multi-body problem, where friction and non-interpenetration constraints are formulated as a linear complementarity problem (LCP) [AE21]. Interactive and real-time applications typically solve the LCPs using iterative fixed-point methods, such as the projected Gauss-Seidel (PGS) method, and efficient methods have been developed for simulating large piles or stacks of bodies [Erl07]. The convergence of iterative solvers can be improved by reordering constraints [AEKT17, PNE10]. Additionally, several strategies have been proposed to parallelize simulations using these solvers. Erleben [Erl04] used graph coloring to solve multi-body systems ignoring interactions between sub-systems. Fratarcangeli et al. [FTP16] similarly proposed an efficient parallel PGS solver based on graph coloring. Other work has proposed splitting techniques for parallelizing large-scale rigid body simulations [IR11].

However, previous studies have noted that iterative solvers can fail to produce realistic behaviour when the simulation requires solving stiff numerical systems (e.g., due to large mass ratios or low compliance [EATK18]). These are the class of simulations targeted by our work. Direct methods are preferred for solving stiff LCPs, such as the block principle pivoting method [JP94] and Lemke's algorithm. This class of methods typically produces more accurate solutions compared to fixed-point iterative methods. However, they are notoriously difficult to parallelize since they often require computing a matrix factorization. Recently, Peiret et al. [PAK*19] proposed a domain decomposition approach that allows stiff multi-body systems to be solved in parallel. Their approach requires partitioning the simulation into smaller sub-systems, along with an interface connecting them. For simulations where a semantic relationship exists between components (e.g., a crane can be divided in components such as a chassis, boom, and cables) there is an obvious *semantic* partitioning. However, for unstructured simulations in which the coupling between simulation bodies can change at each time step, such as a collapsing pile of objects, then an automated technique must be used to partition the simulation. Peiret et al. pro-

DOI: 10.2312/egs.20221036

posed using a minimum degree partitioning that efficiently and automatically computes partitions for unstructured simulations based on the graph of bodies and constraints.

In this work, partitioning algorithms for sub-structured rigid body simulations are examined. The simulations are characterized by contact and thus the constraint graph is not known a priori. We perform extensive comparisons of the MinDP algorithm versus other partitioning strategies, and propose refinements that further increase solver performance.

2 Methodology

A constrained rigid body simulation can be viewed as an undirected graph, where rigid bodies are nodes and constraints between bodies, such as contacts and bilateral joints, are the edges. For instance, graphs for some of our example simulations are visualized in the teaser figure. Note that there is a dual graph where constraints are the nodes and connectivity is determined by the rigid bodies.

2.1 Partitioning Strategies

In this section, several partitioning algorithms are presented for producing contiguous sub-systems that are simulated using the parallel solver proposed by Peiret et al [PAK*19]. Specifically: minimum degree partitioning (MinDP), maximum degree partitioning (MaxDP), breadth-first search (BFS), mass ratio heuristic (MR), and a mixed (MIX) strategy. They are described below.

MinDP [PAK*19] is inspired by Cuthill-McKee equation reordering to produce band-limited Cholesky factorizations.

BFS is a common graph algorithm that we use to generate partitions based on a natural ordering of simulation bodies, incrementally expanding partitions based on constraint connectivity. This algorithm does not consider physical properties.

MaxDP is similar to minimum degree partitioning, but instead expands partitions based on bodies with the highest valence, selecting bodies with more connected constraints.

MR expands an existing partition by examining the mass of adjacent bodies and adding bodies that satisfy

$$\frac{\max\{m_i, m_j\}}{\min\{m_i, m_j\}} < \tau_{MR}\,, \tag{1}$$

where m_i is the current minimum mass of all bodies in the partition, m_j is the mass of the body that will potentially be added to the partition, and τ_{MR} is a user specified mass ratio. The motivation here is to avoid creating poorly conditioned sub-systems and the mass ratio of bodies is contributing factor in this context.

MIX combines MinDP with a partial Fiduccia-Mattheyses algorithm (see Section 2.2). We use a ratio τ_{mix} to decide when to terminate MinDP and switch to adding bodies according to the estimated gain. The pseudocode for this method can be found in Algorithm 1. The algorithm is similar to MinDP, except that the procedure MAXGAINBODY adds bodies with maximal connectivity to the new partition in terms of the number of constraints.

2.2 Refining Strategies

The partitioning algorithms from the previous section focus on finding contiguous groupings of bodies and constraints. However, since the parallel solver used in our experiments is based on a Schur complement domain decomposition technique, there may be benefits to reducing the coupling between sub-systems by reducing the number of edges connecting adjacent partitions. Two graph refinement algorithms that swap bodies at the interface between partitions are evaluated.

Algorithm 1 The mixed partitioning algorithm with parameters $\tau_{\text{mix}} \in (0,1)$ and $n_{\max}$ the maximum bodies per partition.

```
function MIXED(τ_mix, n_max)
    G_0 ← all bodies                          ▷ initialize default partition
    i ← 1                                     ▷ partition id
    while size(G_0) < n_max do
        β ← MINDEGREEBODY(G_0)
        G_i ← {β}
        while size(G_i) < n_max do
            A ← ADJACENTBODIES(G_i) ▷ bodies adjacent to G_i
            if size(G_i) < (τ_mix n_max) then
                β ← MINDEGREEBODY(A ∩ G_0)
            else
                β ← MAXGAINBODY(A ∩ G_0)
            end if
            G_i ← G_i + β                     ▷ add body to the partition
            G_0 ← G_0 − β                     ▷ remove from the default partition
        end while
    end while
end function
```

Kernighan–Lin (KL). The KL algorithm [KL70] uses a heuristic to reduce the number of edges between pairs of partitions. Consider a graph with two partitions – $\mathcal{G}_a$ and $\mathcal{G}_b$. Given body $\beta_i \in \mathcal{G}_a$ and $\beta_j \in \mathcal{G}_b$, we calculate a gain α_{KL} indicating if the edges connecting two partitions will be reduced if the bodies are swapped, such that

$$\alpha_{KL} = (E_i - I_i) + (E_j - I_j) - 2c$$
$$c = \begin{cases} 1 & \text{if } \beta_i \text{ has an edges with } \beta_j \\ 0 & \text{otherwise} \end{cases},$$

where E_i represents the number of edges connecting β_i to $\mathcal{G}_b$ and I_i is the number of edges connecting β_i to $\mathcal{G}_a$. Similarly, E_j is the number of edges connecting β_j to $\mathcal{G}_a$ and I_j is the number of edges connecting β_j to $\mathcal{G}_b$. In our experiments, α_{KL} is computed for every pair of bodies across all pairs of partitions. Pairs of bodies with the maximum positive α_{KL} are then swapped at each iteration; the process is repeated until there is no positive gain.

Fiduccia–Mattheyses (FM). The FM algorithm [FM82] uses a slightly different strategy. Instead of exchanging of a pair of bodies, it only attempts to find one body to swap between partitions. The gain for the FM algorithm is computed as

$$\alpha_{FM} = E_i - I_i\,, \tag{2}$$

where E_i and I_i for body β_i have the same definition as in KL algorithm. Parameter α_{FM} is computed for every node in $\mathcal{G}_a$, and a body is moved to the partition giving the maximum positive α_{FM}. The process is repeated for all pairs of adjacent partitions until neither has a body with positive α_{FM}.

Implementation details. The KL and FM algorithms are mainly suited for optimizing bi-partitioned graphs. For k-way partitioning, a recursive algorithm may be used [KK98]. However, in large-scale rigid body simulations, analyzing all bodies and constraints in the system is time-consuming. Therefore, during the partitioning process, we track peripheral bodies in each partition connected to adjacent partitions. The KL and FM algorithms are then applied to

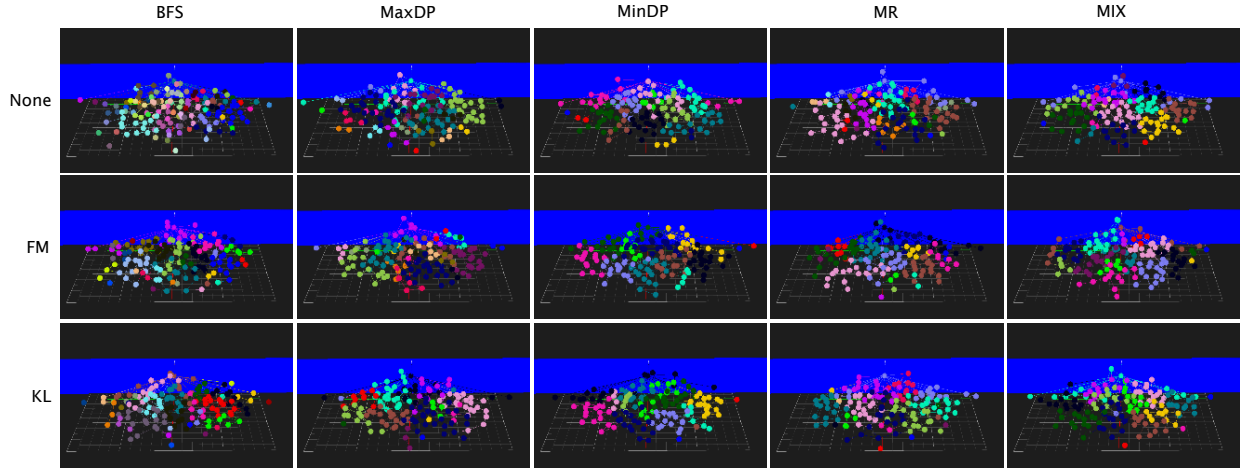

Figure 2: *The partitioning algorithms (columns) and refinements algorithms (rows) applied to the Rock Pile example.*

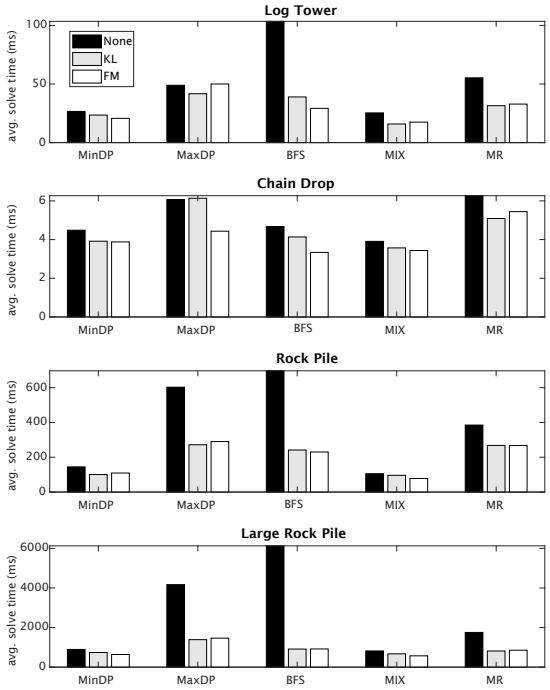

Figure 3: *Average solve time for all examples and combinations of partitioning and refinement algorithms.*

bodies at the interface between partitions, with gains of bodies at the interface between adjacent partitions being computed.

3 Results

Simulations were performed using the Vortex physics engine and Intel Core i9 2.40GHz CPU with 8 cores (16 threads). Partitioning and refinement algorithms were implemented as a C++ plugin.

We evaluate the partitioning algorithms by reproducing several examples from the work by Peiret et al. [PAK*19]: Chain Drop, Log Tower, and Rock Pile. We also introduce a second variation of the last example, Large Rock Pile, which has three times as many rocks with random masses of 10 kg, 100 kg, and 1000 kg.

Figure 2 shows the sub-systems produced by the partitioning and refinement algorithms for a selected frame from the Rock Pile example. Bodies in the same partition are visualized as spheres of the same color (see supplementary video).

3.1 Solver Performance

Performance is evaluated by computing the solve time over the first 500 time steps of each simulation. Figure 3 summarizes these statistics for each of the five partitioning strategies with i) no refinement, ii) KL refinement, and iii) FM refinement. In all examples, the MinDP and MIX partitioning algorithms gave the best performance, with the refinement algorithms giving an additional performance boost. For instance, in the Log Tower example, FM and KL reduced solve time by about 31% and 12%, respectively, compared to the MinDP algorithm without refinement. Max solve times were also reduced (see Figure 5). The MIX algorithm with FM refinement also reduced the solve time by 46% and 35% for the Rock Pile and Large Rock Pile examples. We also observed that performance gains do not depend on problem size. Figure 4 shows the average solve time for different scales of the Rock Pile example (50 to 800 bodies). The FM and KL refinement consistently reduce solver time. However, significant performance improvements were not observed for all examples. For instance in the Chain Drop example, the simplest scenario we tested, the MinDP resulted in 4 ms solve time on average, and KL and FM gave only limited improvements. Complete timing information for the solver and partitioning methods can be found in Table 1. The MIX algorithm with KL or FM refinement typically gave best performance, except for the Chain Drop example where the BFS algorithm with FM refinement gave a better result.

4 Conclusions

Solving multi-body systems with stiff constraints and contact is non-trivial. The MinDP proposed by Peiret et al. for their parallel sub-structuring solver gives good performance compared to other strategies, and applying the KL and FM algorithms to refine partitions can further improve solver performance. We have additionally proposed a new partitioning algorithm, called MIX, that gives the lowest solver wall clock time across all examples we tested, not accounting for overhead due to partitioning. We believe that computational overhead of the partitioning and refinement algorithms could be significantly improved with careful optimization of the code, which would further reduce timings reported in Table 1. It would be interesting to experiment with min-cut type partitioning algo-

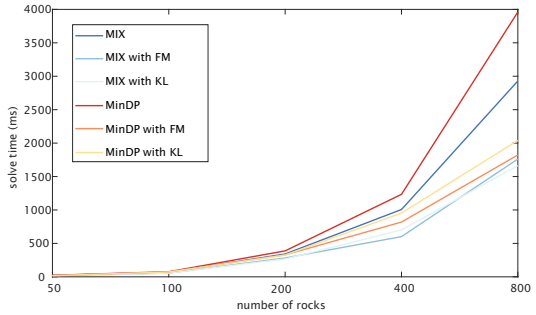

Figure 4: *Average solve times for the Rock Pile example for different number of rocks. The KL and FM refinement algorithms improve the performance, even for different system sizes.*

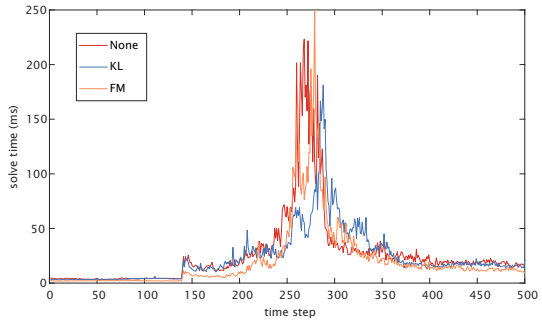

Figure 5: *The solve time per time step for the Log Tower when MinDP is used. FM and KL refinements improve performance and help to reduce peaks.*

rithms. However, preliminary tests we performed using Karger's algorithm [Kar93] were not competitive with the methods evaluated in this paper. Finally, our simulation experiments target a parallel sub-structuring solver using a direct numerical method, making it well suited to stiff and poorly conditioned system. However, other partitioning strategies may be better suited for iterative solvers.

Acknowledgments

This work was supported by an NSERC Collaborative Research and Development (CRD) grant and a Mitacs Globalink Graduate Fellowship. We thank Chantal Hutchison and Joe Hewlett for their feedback and stimulating discussions, and Andreas Enzenhoefer for his help preparing the simulations.

References

[AE21] ANDREWS S., ERLEBEN K.: Contact and friction simulation for computer graphics. In *ACM SIGGRAPH 2021 Courses* (2021), ACM. 1

[AEKT17] ANDREWS S., ERLEBEN K., KRY P. G., TEICHMANN M.: Constraint reordering for iterative multi-body simulation with contact. In *Proc. of ECCOMAS Thematic Conf. on Multibody Dynamic* (2017), pp. 18–22. 1

[EATK18] ENZENHÖFER A., ANDREWS S., TEICHMANN M., KÖVECSES J.: Comparison of Mixed Linear Complementarity Problem Solvers for Multibody Simulations with Contact. In *Proc. of VRIPHYS 2018* (2018), The Eurographics Association. 1

[Erl04] ERLEBEN K.: *Contact Graphs in Multibody Dynamics Simulation*. Tech. rep., University of Copenhagen, 2004. 1

[Erl07] ERLEBEN K.: Velocity-based shock propagation for multibody dynamics animation. *ACM Trans. Graph. 26*, 2 (jun 2007), 12–es. 1

[FM82] FIDUCCIA C. M., MATTHEYSES R. M.: A linear-time heuristic for improving network partitions. In *19th Design Automation Conf.* (1982), IEEE, pp. 175–181. 2

Table 1: *The average solve time, t_s, and partitioning & refinement time, t_{pr}, for all the examples in ms. The lowest combined solve time plus partitioning times for each example are highlighted in yellow.*

	MinDP					
	None		**KL**		**FM**	
	t_s	t_{pr}	t_s	t_{pr}	t_s	t_{pr}
Log Tower	26.6	10.8	23.5	14.7	20.7	15.1
Chain Drop	4.5	1.6	3.9	1.7	3.9	2.1
Rock Pile	144.5	1.2	100.7	1.7	109.2	1.7
Large Rock Pile	894.1	5.5	738.2	7.7	638.7	5.6
	MaxDP					
	None		**KL**		**FM**	
	t_s	t_{pr}	t_s	t_{pr}	t_s	t_{pr}
Log Tower	48.9	5.0	41.7	11.3	50.0	8.1
Chain Drop	6.1	0.3	6.1	0.8	4.4	0.4
Rock Pile	602.8	1.8	271.9	4.6	290.3	3.0
Large Rock Pile	4166.8	4.3	1387.7	10.6	1463.3	6.4
	BFS					
	None		**KL**		**FM**	
	t_s	t_{pr}	t_s	t_{pr}	t_s	t_{pr}
Log Tower	103.4	0.6	39.0	5.2	29.2	2.3
Chain Drop	4.7	0.2	4.1	0.6	3.3	0.3
Rock Pile	697.1	0.6	241.6	4.4	230.2	2.8
Large Rock Pile	6127.1	0.9	912.5	8.3	918.4	3.2
	MR					
	None		**KL**		**FM**	
	t_s	t_{pr}	t_s	t_{pr}	t_s	t_{pr}
Log Tower	55.3	0.8	31.5	7.5	32.9	3.4
Chain Drop	6.3	0.2	5.1	0. 6	5.4	0.4
Rock Pile	385.0	0.7	268.1	3.4	267.2	1.8
Large Rock Pile	1752.6	1.2	814.3	5.1	854.5	2.6
	MIX					
	None		**KL**		**FM**	
	t_s	t_{pr}	t_s	t_{pr}	t_s	t_{pr}
Log Tower	25.3	9.5	15.9	12.7	17.5	9.6
Chain Drop	3.9	1.3	3.6	1.7	3.4	2.4
Rock Pile	105.3	1.1	96.1	1.7	78.1	1.5
Large Rock Pile	816.2	4.7	671.1	7.1	570.3	5.8

[FTP16] FRATARCANGELI M., TIBALDO V., PELLACINI F.: Vivace: A practical gauss-seidel method for stable soft body dynamics. *ACM Trans. Graph. 35*, 6 (2016), 1–9. 1

[IR11] IGLBERGER K., RÜDE U.: Large-scale rigid body simulations. *Multibody system dynamics 25*, 1 (2011), 81–95. 1

[JP94] JÚDICE J. J., PIRES F. M.: A block principal pivoting alg. for large-scale strictly monotone linear complementarity problems. *Computers & Operations Research 21*, 5 (1994), 587–596. 1

[Kar93] KARGER D. R.: Global min-cuts in rnc, and other ramifications of a simple min-cut alg. In *Proc. of the ACM-SIAM Symp. on Discrete Algs.* (1993), Society for Industrial and Appl. Maths., pp. 21—-30. 4

[KK98] KARYPIS G., KUMAR V.: A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing 20*, 1 (1998), 359–392. 2

[KL70] KERNIGHAN B. W., LIN S.: An efficient heuristic proc. for partitioning graphs. *The Bell Sys. Tech. Journal 49*, 2 (1970), 291–307. 2

[PAK*19] PEIRET A., ANDREWS S., KÖVECSES J., KRY P. G., TEICHMANN M.: Schur complement-based substructuring of stiff multibody systems with contact. *ACM Trans. Graph. 38*, 5 (oct 2019). 1, 2, 3

[PNE10] POULSEN M., NIEBE S., ERLEBEN K.: Heuristic convergence rate improvements of the projected gauss-seidel method for frictional contact problems. In *Proc. of WSCG 2010* (2010). 1