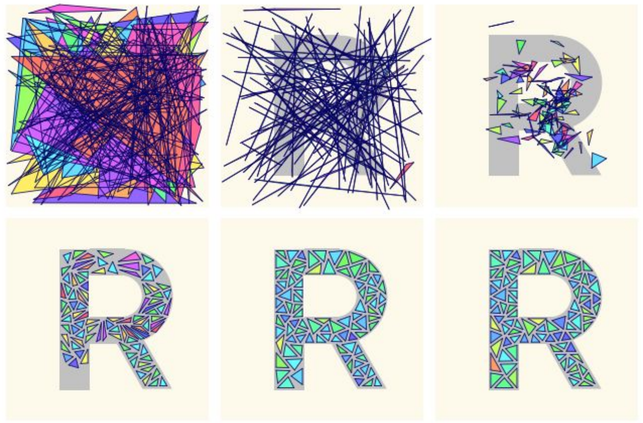
Minkowski Penalties: Robust Differentiable Constraint Enforcement for Vector Graphics

Authors: Jiří Minarčík, Sam Estep, Wode Ni, Keenan CraneAuthors Info & Claims

SIGGRAPH Conference Papers ’24, July 27–August 01, 2024

INTRODUCTION

- This paper describes an optimization-based framework for finding arrangements of 2D shapes subject to pairwise constraints.
- We approach this problem through the minimization of novel energetic penalties, derived from the signed distance function of the Minkowski difference between interacting shapes.
- initialized from a wildly infeasible state, and, unlike many common collision penalties, can handle open curves that do not have a well-defined inside and outside.
- it supports rich features beyond the basic no-overlap condition, such as tangency, containment, and precise padding, which are especially valuable in the vector illustration context.



RELATED WORK

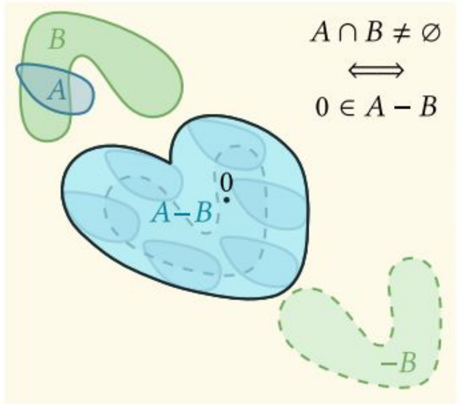
- Robotics and Motion Planning:
    - In robotics, penetration depth is commonly used to resolve or prevent collisions in a dynamical context [Kim et al. 2002]. For instance, Minkowski differences are used to detect interpenetration [Kockara et al. 2007], or suggest a direction for contact resolution [Dobkin et al. 1993], but we did not find work that directly differentiates penetration depth.
    - Conversely, recent work on differentiable collision detection/resolution does not use Minkowski-based penalties [Zimmermann et al. 2022], and is limited to convex geometry [Tracy et al. 2023; Montaut et al. 2023].
- Physical Simulation and Geometry Processing:
    - Since the objective is to find a non-intersecting state, most of these methods consider only feasible initialization, often using domain-specific knowledge [Smith and Schaefer 2015]. These methods also consider only volumetric domains that have an inside/outside [Bridson et al. 2005]

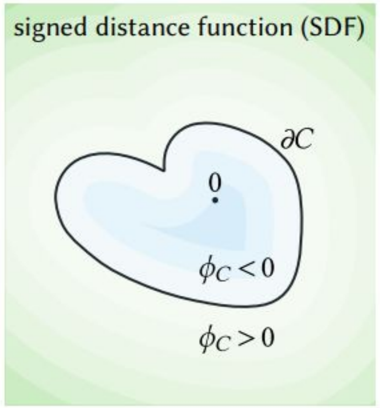
BACKGROUND

- Minkowski Difference

The *Minkowski sum* $A + B$ of any two sets $A, B \subset \mathbb{R}^n$ is the set

$$A + B := \{\mathbf{a} + \mathbf{b} \colon \mathbf{a} \in A, \mathbf{b} \in B\}.$$





- Signed Distance Functions

For any n-dimensional region $A \subset \mathbb{R}^n$, let

$$d(\mathbf{x}, \partial A) := \min_{\mathbf{y} \in \partial A} |\mathbf{x} - \mathbf{y}|$$

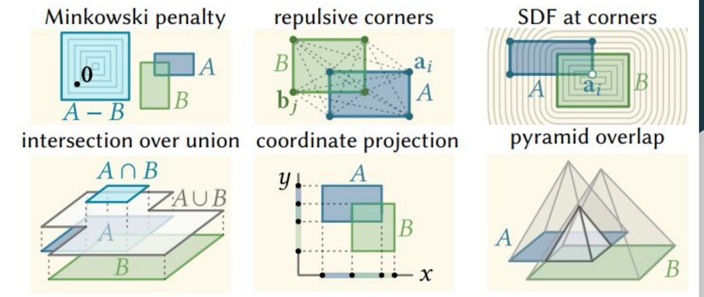
denote the (unsigned) distance from any point $\mathbf{x}$ to the closest point on A's boundary. The *signed distance function (SDF)* for A is then

$$\phi_A(\mathbf{x}) := \begin{cases} -d(\mathbf{x}, \partial A) & \mathbf{x} \in A, \\ d(\mathbf{x}, \partial A) & \mathbf{x} \notin A. \end{cases}$$

PENALTIES

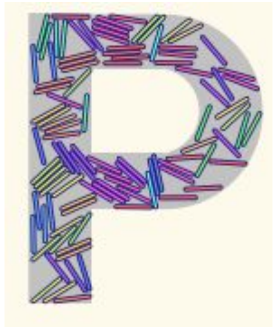
Rectangle Penalties:

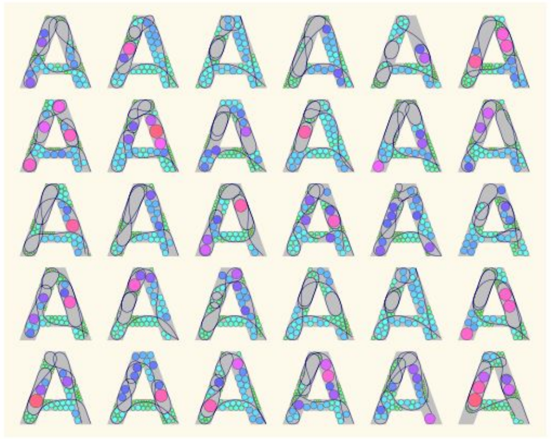
- Intersection over Union: The overlap area $|A \cap B|$ normalized by total area $|A \cup B| = |A| + |B| - |A \cap B|$—known as intersection over union [Rezatofighi et al. 2019]. Here the gradient can be zero even when A and B overlap (e.g., they form a “cross”, or $A \subset B$).
- Coordinate Projection: The minimum of the horizontal and vertical range of overlap, which is nonzero if $A \cap B \neq \varnothing$. Again, the gradient can be zero in, e.g., nested or “cross” configurations.
- Repulsive Corners: The sum of Coulomb potentials $1/|\mathrm{a}i - \mathrm{b}j\,|$ over all pairs of rectangle corners $\mathrm{a}i$, $\mathrm{b}j$. Here, a balance of Coulomb forces can still yield overlapping configurations.
- SDF at Corners: The sum of $\min(0, -\phi A\ (\mathrm{b}i))$ over all corners of B and likewise for A, which is zero if all corners of A are outside B and vice versa. Here, rectangles can again overlap in a “cross.”
- Pyramid Overlap: Consider a pyramid over each rectangle, with height inversely proportional to area. Since tall, narrow pyramids “pierce through” large, flat ones, their overlap volume has a nonzero gradient for intersecting configurations [Jacobson 2021]. However, this volume is hard to differentiate (we use finite differences of mesh booleans), and does not apply to general shapes.



Approach	(Min)	(Diff)	(Grad)	(Shape)	(Param)
Intersection over union	✓	✓	·	·	✓
Coordinate projection	✓	✓	·	·	·
Repulsive corners	✓	✓	·	✓	✓
SDF at corners	·	✓	·	·	✓
Pyramid overlap	✓	·	✓	·	✓
Minkowski penalty	✓	✓	✓	✓	✓

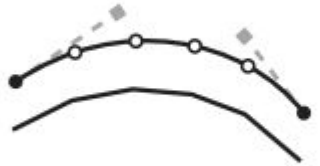
- Open Curves: An open curve γ (like a line segment or hemicircle) does not have a well-defined “inside” and “outside.”





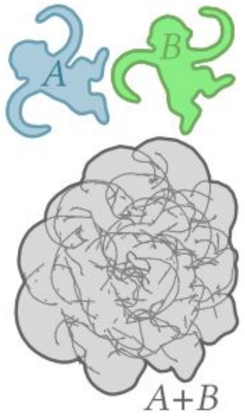
- Padding: A common case is accounting for stroke width, which effectively pads the original shape. A benefit of our SDF-based formulation is that we achieve exact padding by simply adding or subtracting w from the penetration depth ϕC (0)

- Polygonal Approximation : Prior to computing Minkowski differences, we approximate all Bézier curves by polygons.

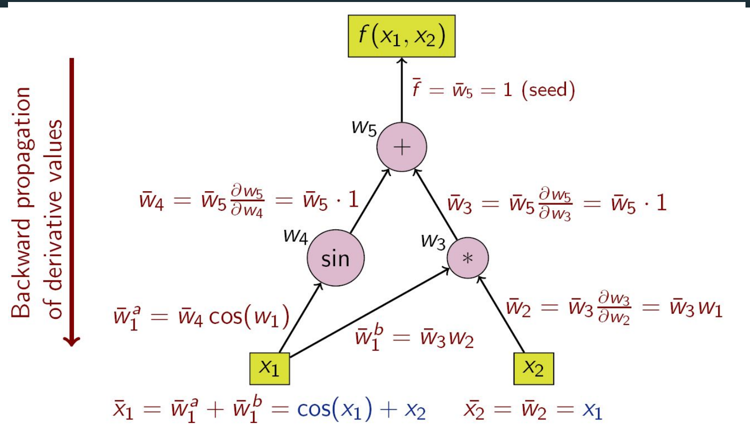


- both partitioning and polygon union can be hard to implement efficiently and robustly. We instead opt for convolution, which is efficient in practice, and comes with a wellestablished theory .

- Minkowski Difference of Polygons :
    - More generally, for simple nonconvex polygons there are two basic methods: decomposition and convolution [Wein et al. 2023]. Decomposition partitions A and B into convex regions, takes all pairwise Minkowski sums of these regions, and merges the sums [Margalit and Knott 1989]. However, both partitioning and polygon union can be hard to implement efficiently and robustly [Behar and Lien 2011].
- We instead opt for convolution



OPTIMIZATION



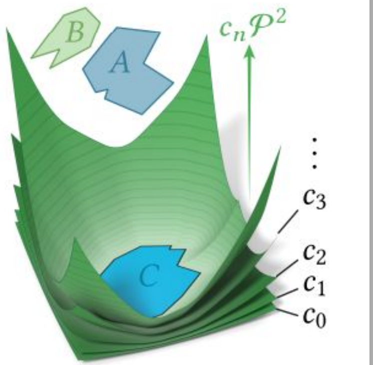
$f(x_1, x_2)$
$\bar{f} = \bar{w}_5 = 1$ (seed)
w_5
$+$
$\bar{w}_4 = \bar{w}_5 \frac{\partial w_5}{\partial w_4} = \bar{w}_5 \cdot 1$
$\bar{w}_3 = \bar{w}_5 \frac{\partial w_5}{\partial w_3} = \bar{w}_5 \cdot 1$
w_4
sin
w_3
$*$
Backward propagation of derivative values
$\bar{w}_1^a = \bar{w}_4 \cos(w_1)$
$\bar{w}_1^b = \bar{w}_3 w_2$
$\bar{w}_2 = \bar{w}_3 \frac{\partial w_3}{\partial w_2} = \bar{w}_3 w_1$
x_1
x_2
$\bar{x}_1 = \bar{w}_1^a + \bar{w}_1^b = \cos(x_1) + x_2$
$\bar{x}_2 = \bar{w}_2 = x_1$

- Automatic Differentiation:
  - use reverse-mode autodiff [Speelpenning 1980] to differentiate penalties
  - Tools from machine learning, like PyTorch [Paszke et al. 2019], focus on relatively simple tensor-based computation and are ill-suited to our scalar- and algorithm-heavy penalty functions.
  - Here, more conventional engines such as Zygote [Innes et al. 2019] and Enzyme [Moses and Churavy 2020] are more suitable.
  - In particular, since we target web-based applications , we use Rose [Estep et al. 2024], which runs natively in the browser (and was two orders of magnitude faster than Zygote).

Exterior Point Method:

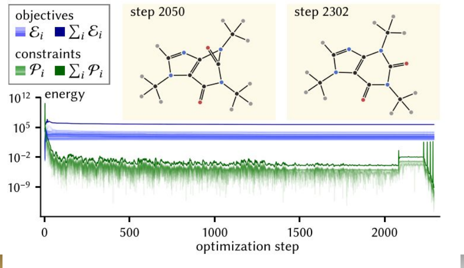
- Following a strategy suggested by Ye et al. [2020], we adopt an exterior point method to solve
- This method permits infeasable initialization, via progressive stiffening of constraints.

$$\min_{\mathbf{p}\in\mathbb{R}^m} \sum_{i=1}^{k} \mathcal{E}(\mathbf{p}) + c_n \sum_{i=1}^{l} \mathcal{P}_i^2(\mathbf{p}), \quad n = 0, 1, 2, \cdots \tag{2}$$



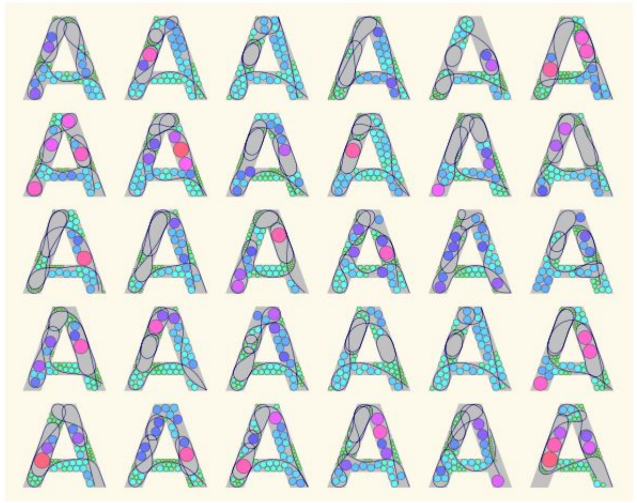
Guarantees and Failure Modes:

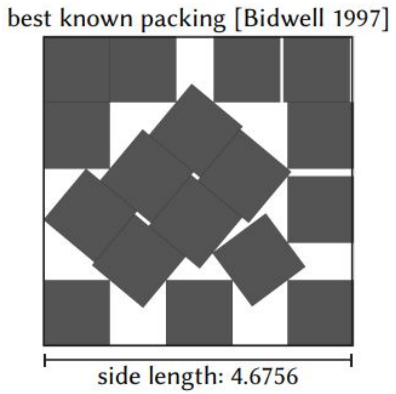
- Suppose, for instance, we want to enforce the no-overlap predicate $A \cap B = \varnothing$, via the penalty Pd (A, B)
- For simplicity, imagine that A is fixed, and B can only translate by an offset p ∈ R2
- we get a nonzero gradient $|\nabla \text{p}\phi C| = 1$ whenever A and B overlap
- Moreover, since the gradient norm is bounded away from zero (namely, always equal to 1), we cannot have an infeasible limit poin

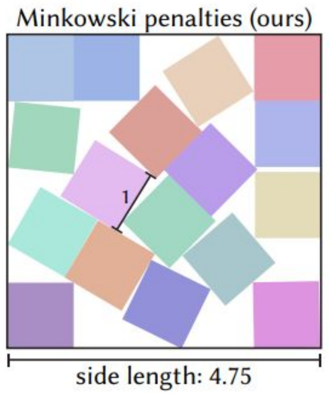


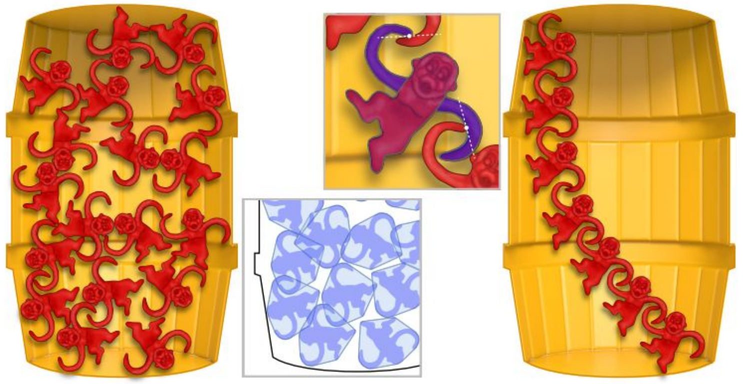
RESULTS AND EVALUATION

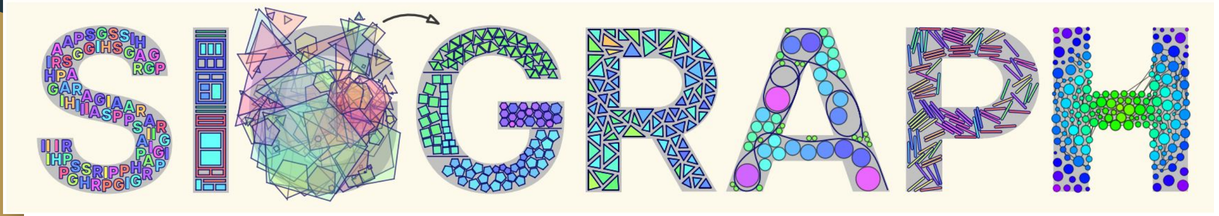
- In this section we evaluate the effectiveness of our approach and compare it to alternative methods
-

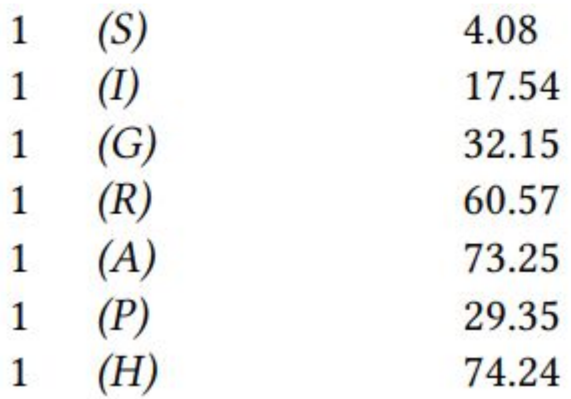




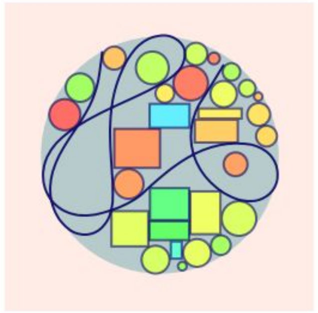


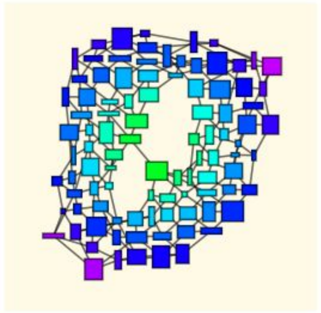




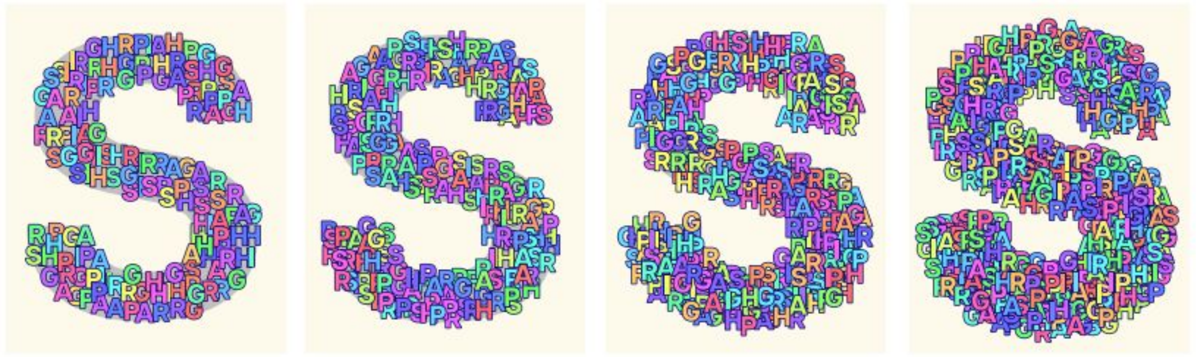


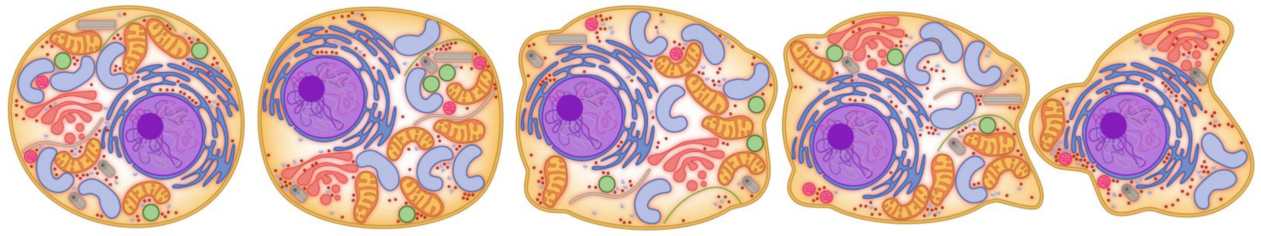
1	*(S)*	4.08
1	*(I)*	17.54
1	*(G)*	32.15
1	*(R)*	60.57
1	*(A)*	73.25
1	*(P)*	29.35
1	*(H)*	74.24



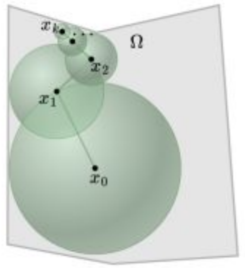
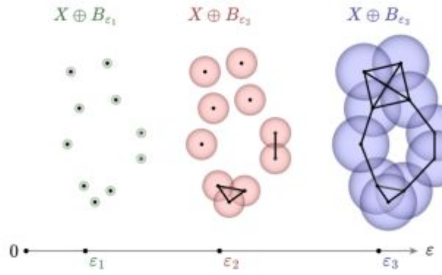
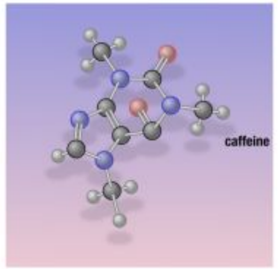


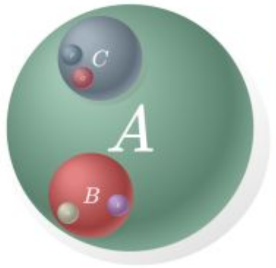
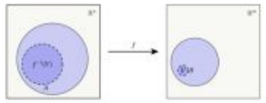
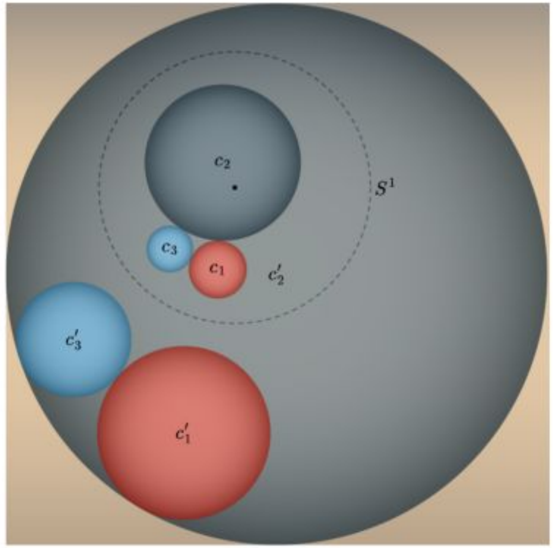
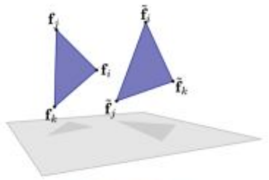
Verdana
Rockwell
Courier
Gotham
Helvetica
Lobster
Times
Didot
Bodoni
Caslon
Baskerville
no overlap
Comic Sans
DIN
Frutiger
Futura
Avant Garde
Arial
Palatino
Calibri
Garamond
Georgia
Impact





LIMITATIONS AND FUTURE WORK


x_k
x_2
x_1
x_0
Ω

$X \oplus B_{\varepsilon_1}$
$X \oplus B_{\varepsilon_2}$
$X \oplus B_{\varepsilon_3}$
0
ε_1
ε_2
ε_3
ε

caffeine


C
A
B


c_2
S^1
c_3
c_1
c_2'
c_3'
c_1'

$\mathbf{f}_j$
$\mathbf{f}_i$
$\mathbf{f}_k$
$\tilde{\mathbf{f}}_i$
$\tilde{\mathbf{f}}_k$
$\tilde{\mathbf{f}}_j$