

開発者の担当可能なタスク量を考慮した バグトリージ手法の提案

和歌山大学大学院システム研究科 博士後期課程
日本学術振興会 特別研究員（DC）

柏祐太郎

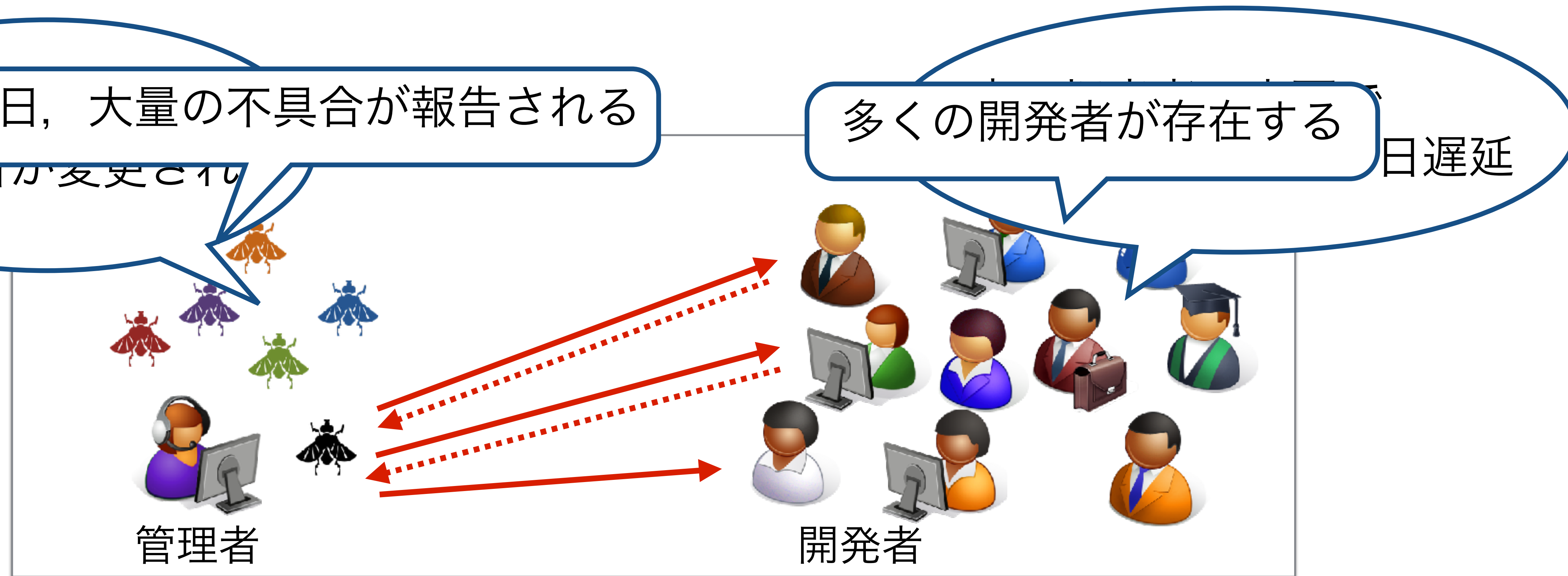
現状のタスク割当て手法と問題点

2

管理者が不具合報告の内容を精査し、担当の開発者を決定

44% 毎日、大量の不具合が報告される
担当者が変更される

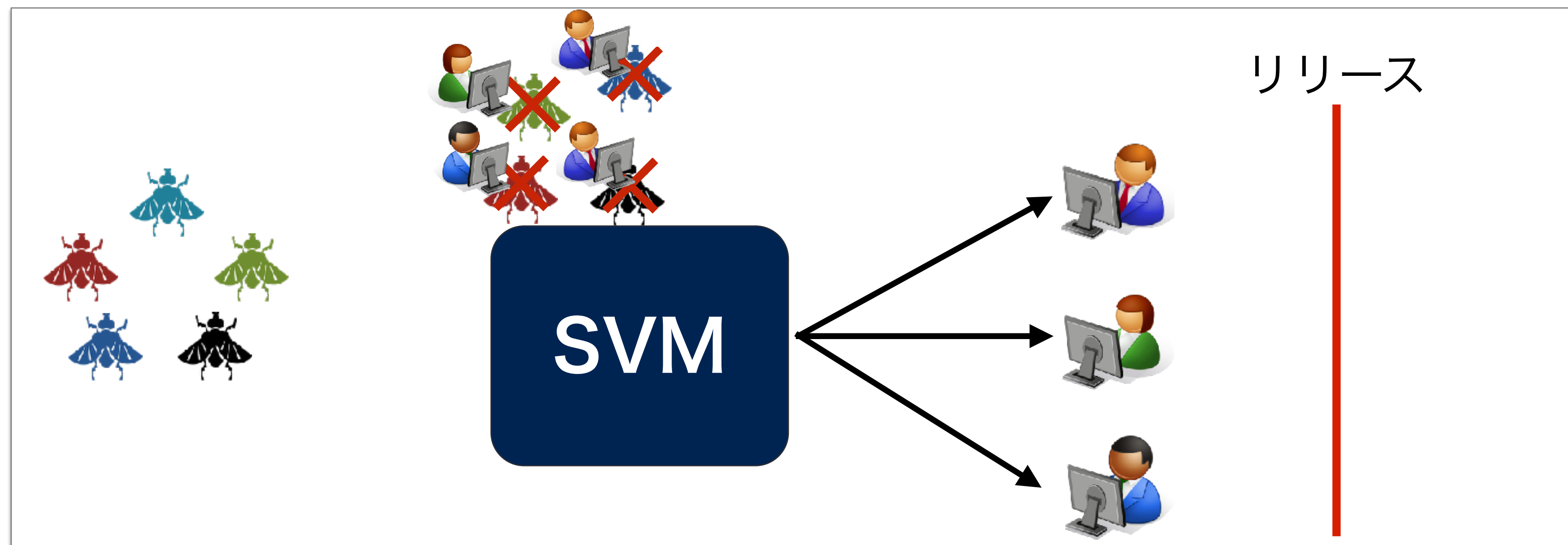
多くの開発者が存在する
日遅延



不具合の担当者変更（再割当）の頻発により不具合修正が長期化

既存手法とその問題点

機械学習（SVM）を用いたタスク割当て



メリット：

○ 適任の開発者に割当てることができる

デメリット：

× 一部の開発者にタスクが集中する傾向

本研究の着眼点

有能な開発者でも修正可能な不具合数は限られている

- 多くの不具合がリリースに間に合わない可能性
- タスクの集中は再割当の主な原因のひとつ

→ **担当可能なタスク量**を考慮することが必要

修正の効率	各開発者	プロジェクト
既存手法	◎	×
提案手法	○	◎

各開発者の効率が低下することのある程度許容し、
プロジェクト全体の効率を上げることを狙う

本研究の目的とアプローチ

目的:

プロジェクト全体で修正効率が良くなる割当て手法を構築

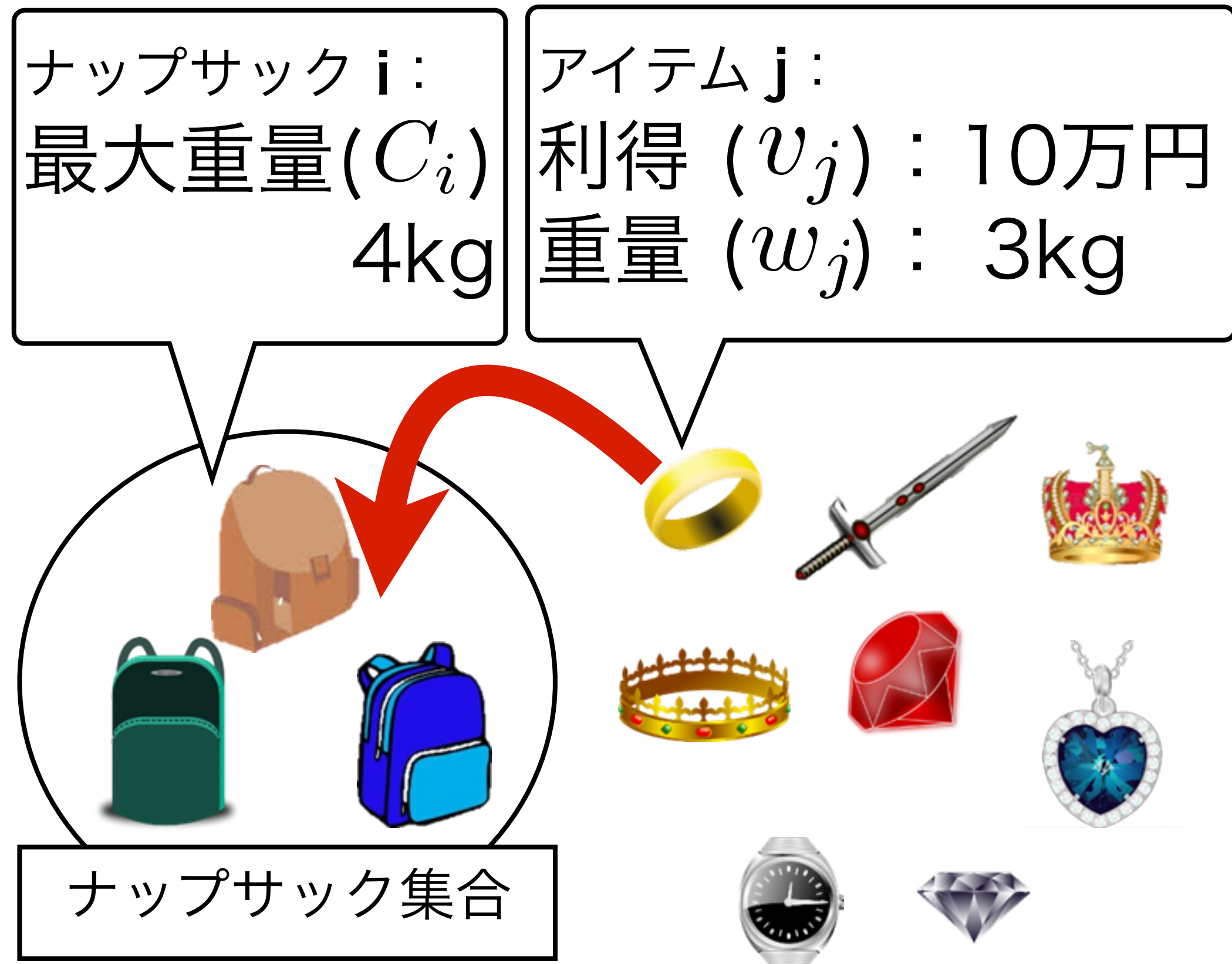
- 個々の修正タスクに対する開発者の適性を考慮
- 一定期間に**担当可能なタスク量**を考慮

アプローチ:

不具合の割当問題を**マルチナップサック問題**とみなして,
不具合と開発者の最適な組合せを求める

マルチナップサック問題

ナップサック集合に入れるアイテムの
総利得を最大にするアイテムの組合せを求める問題



目的関数

$$\text{Maximize : } \sum_{i=1}^m \sum_{j=1}^n v_j x_{ij}$$

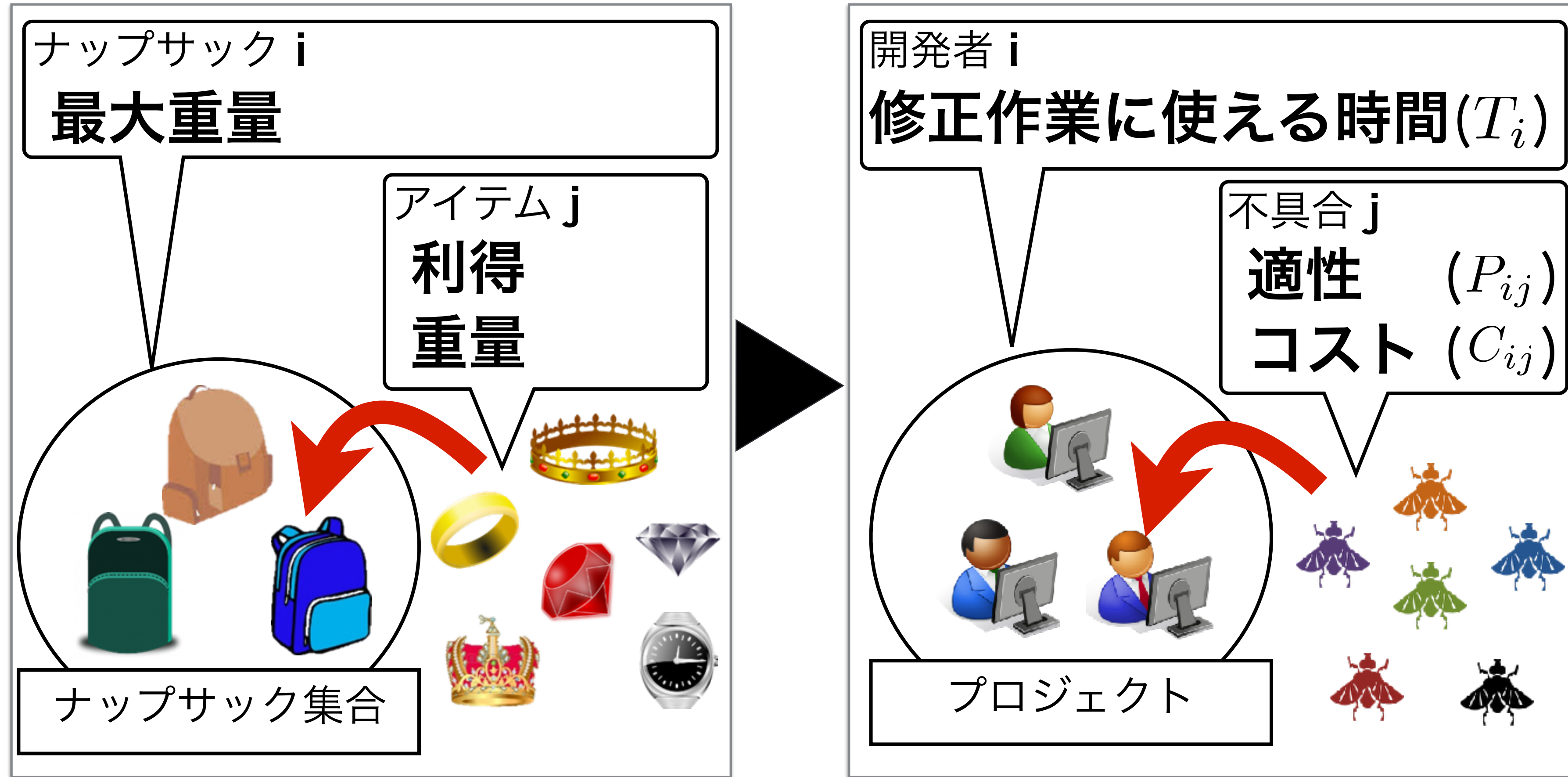
$$x_{ij} \in \{0, 1\}$$

制約条件

$$\sum_{j=1}^n w_j x_{ij} \leq C_i \quad (i = 1, \dots, m)$$

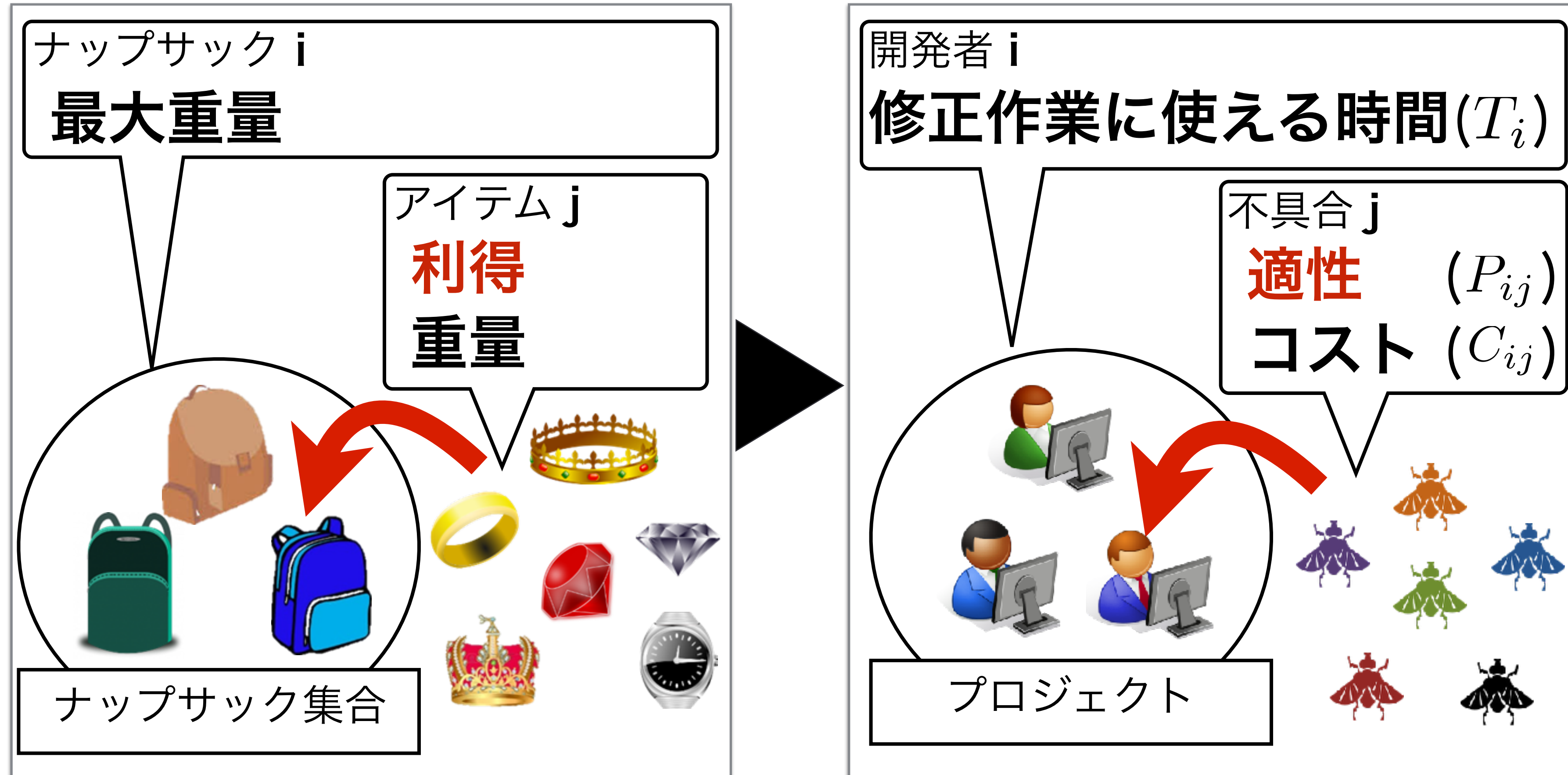
$$\sum_{i=1}^m x_{ij} \leq 1 \quad (j = 1, \dots, n)$$

キーアイデア



プロジェクト全体で修正効率が最も良くなる
不具合と開発者の組合せを求める

キーアイデア

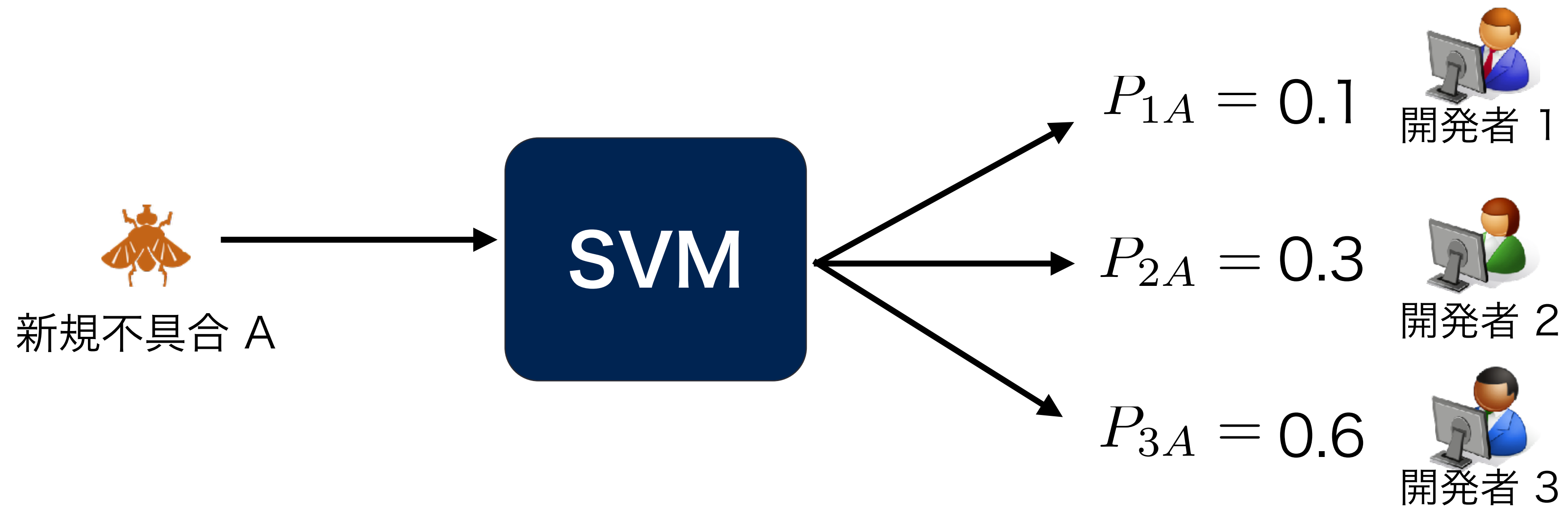


プロジェクト全体で修正効率が最も良くなる
不具合と開発者の組合せを求める

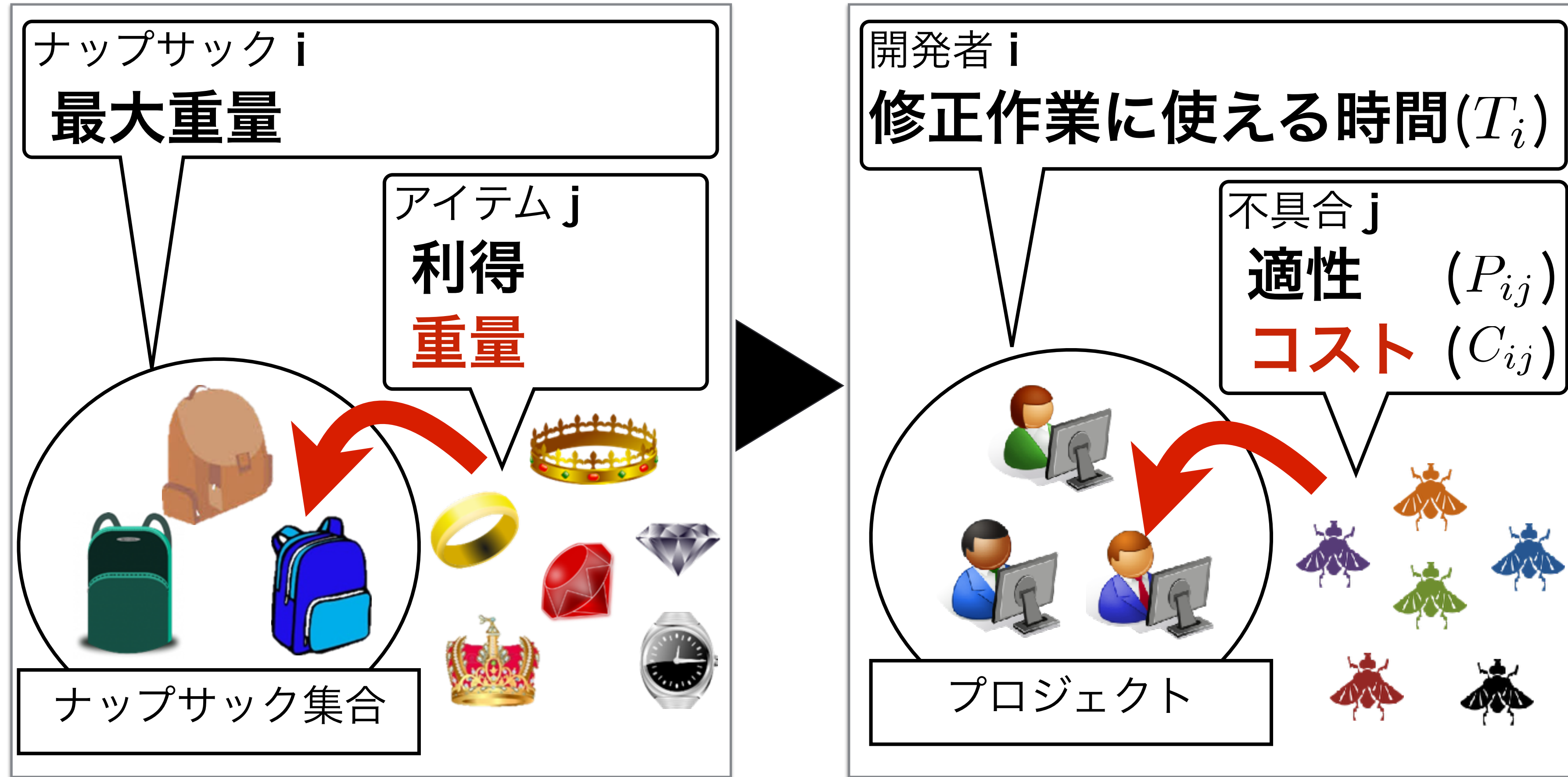
適性（プリファレンス P_{ij} ）の算出方法

P_{ij} とは開発者 i が不具合 j の修正に適任である確率

- 既存手法と同じ方法（SVMを使用）
- 不具合報告の内容から開発者を学習



キーアイデア

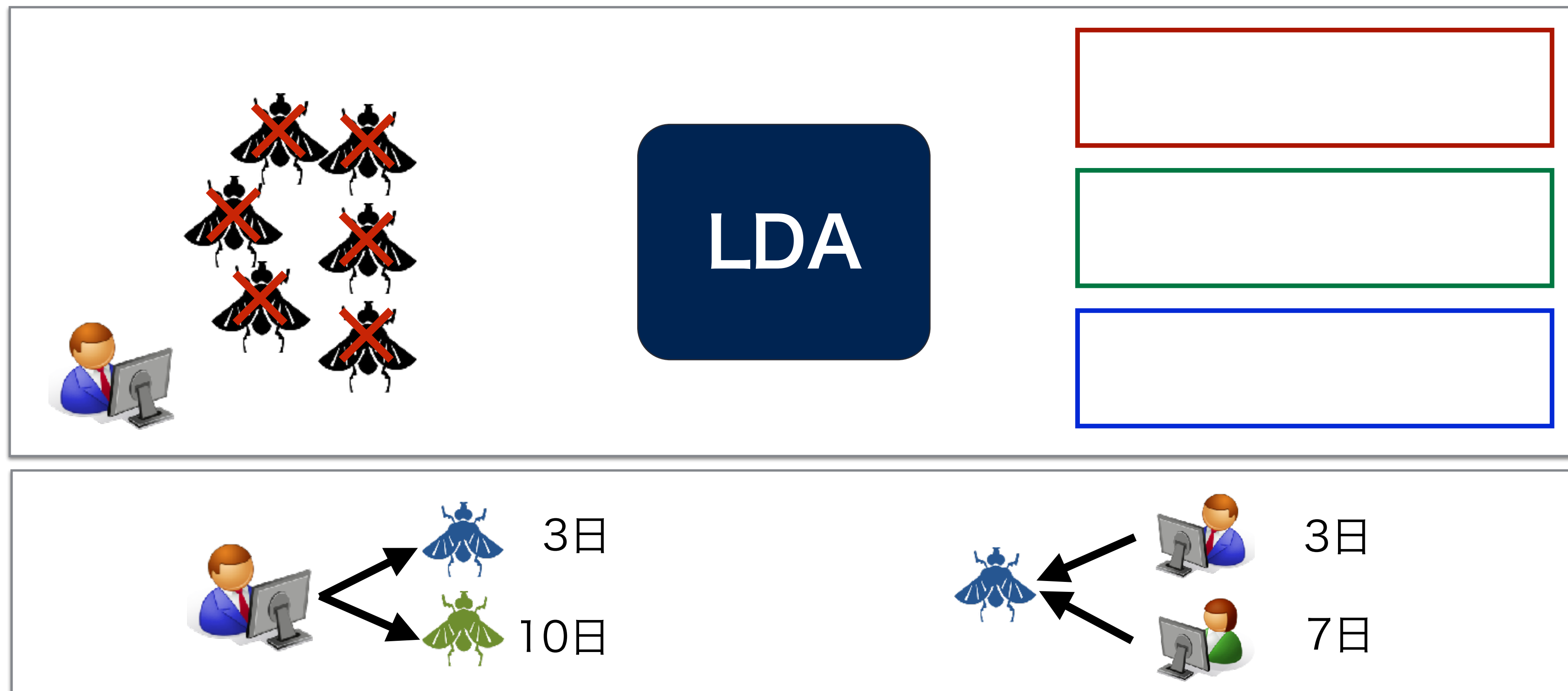


プロジェクト全体で修正効率が最も良くなる
不具合と開発者の組合せを求める

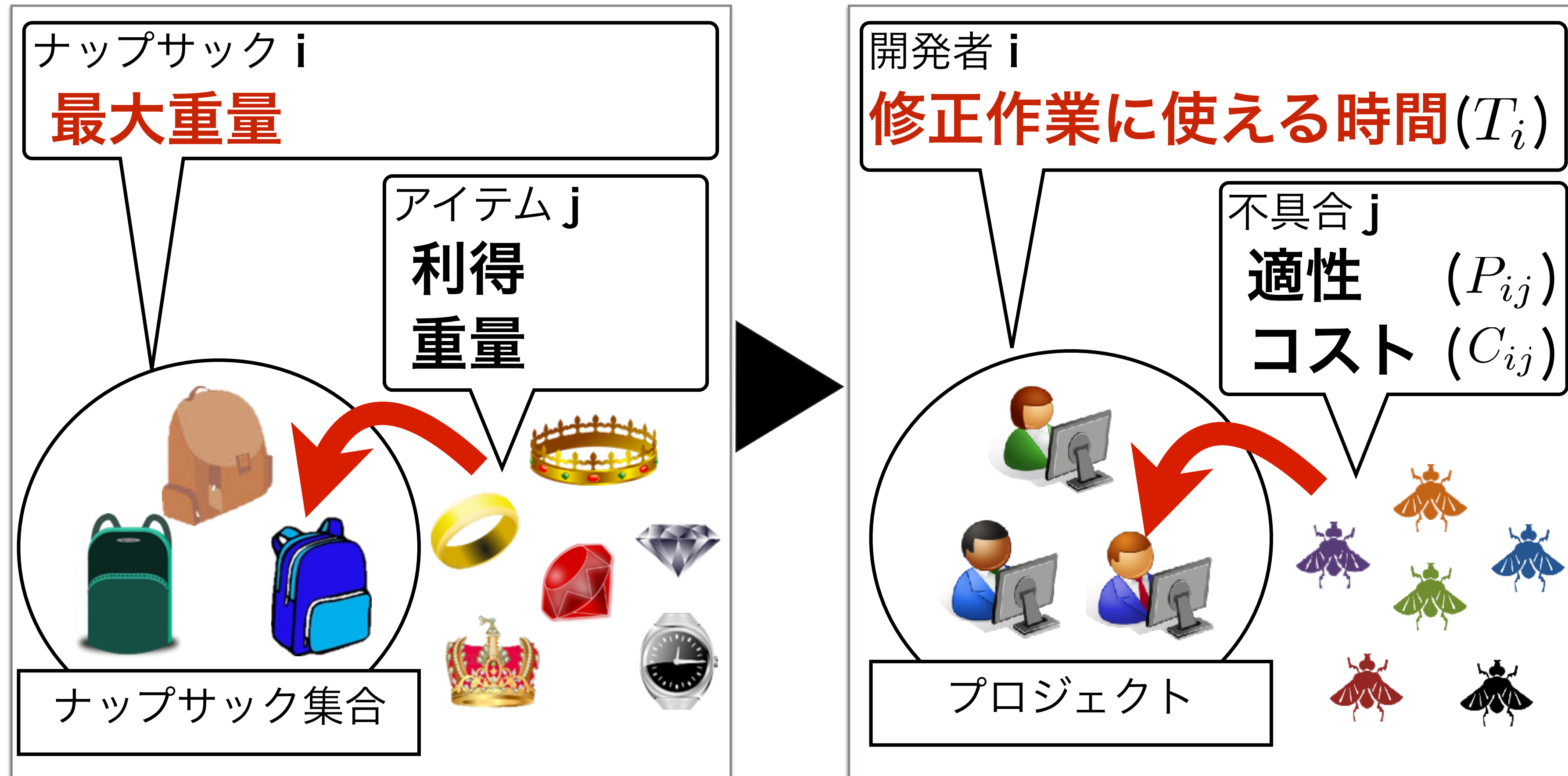
コストの算出方法

C_{ij} とは開発者 i が不具合 j の修正に必要な時間

- Latent Dirichlet Allocationを用いて不具合を分類
- 分類ごとの不具合修正時間の平均値をコストとする



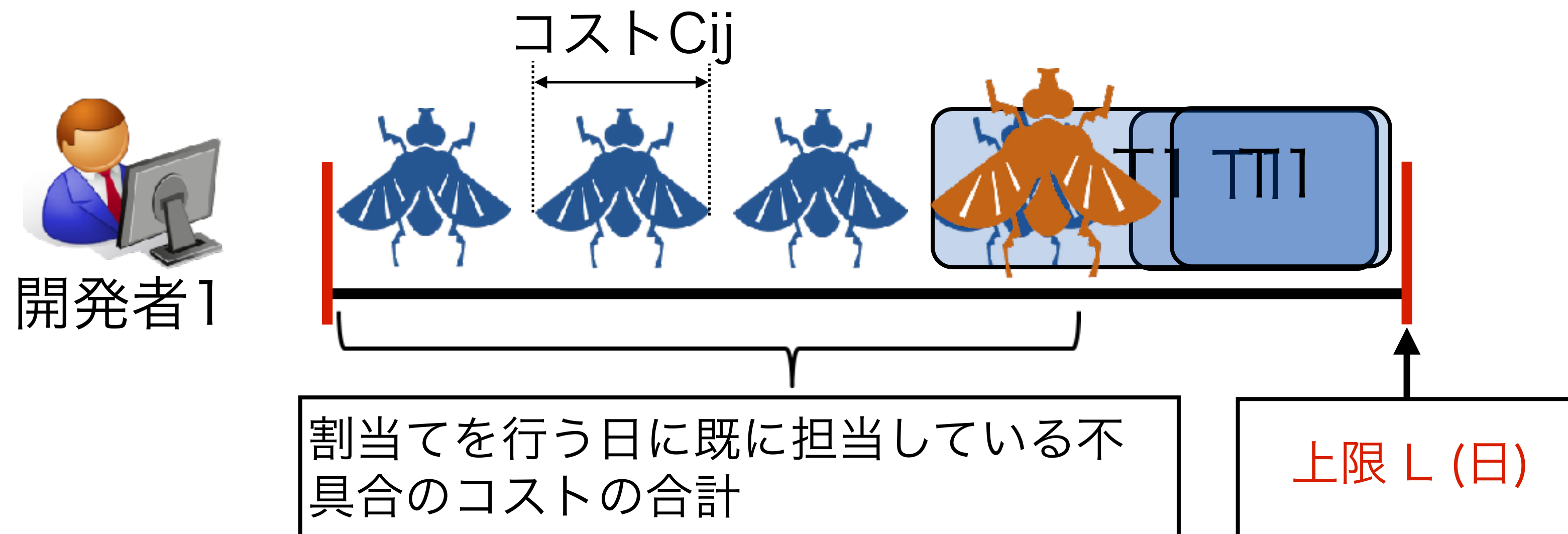
キーアイデア



プロジェクト全体で修正効率が最も良くなる
不具合と開発者の組合せを求める

修正作業に使える時間 (Ti)

$$T_i = \text{上限 } L - \sum_{j=1}^n C_{ij} x_{ij} \quad (i = 1, \dots, m)$$



定式化

目的関数：

プロジェクト全体で，プリファレンスの総和が最大になる様に開発者にタスクを割当てる

$$\text{Maximize : } \sum_{i=1}^m \sum_{j=1}^n P_{ij} x_{ij}$$

$$x_{ij} \in \{0, 1\}$$

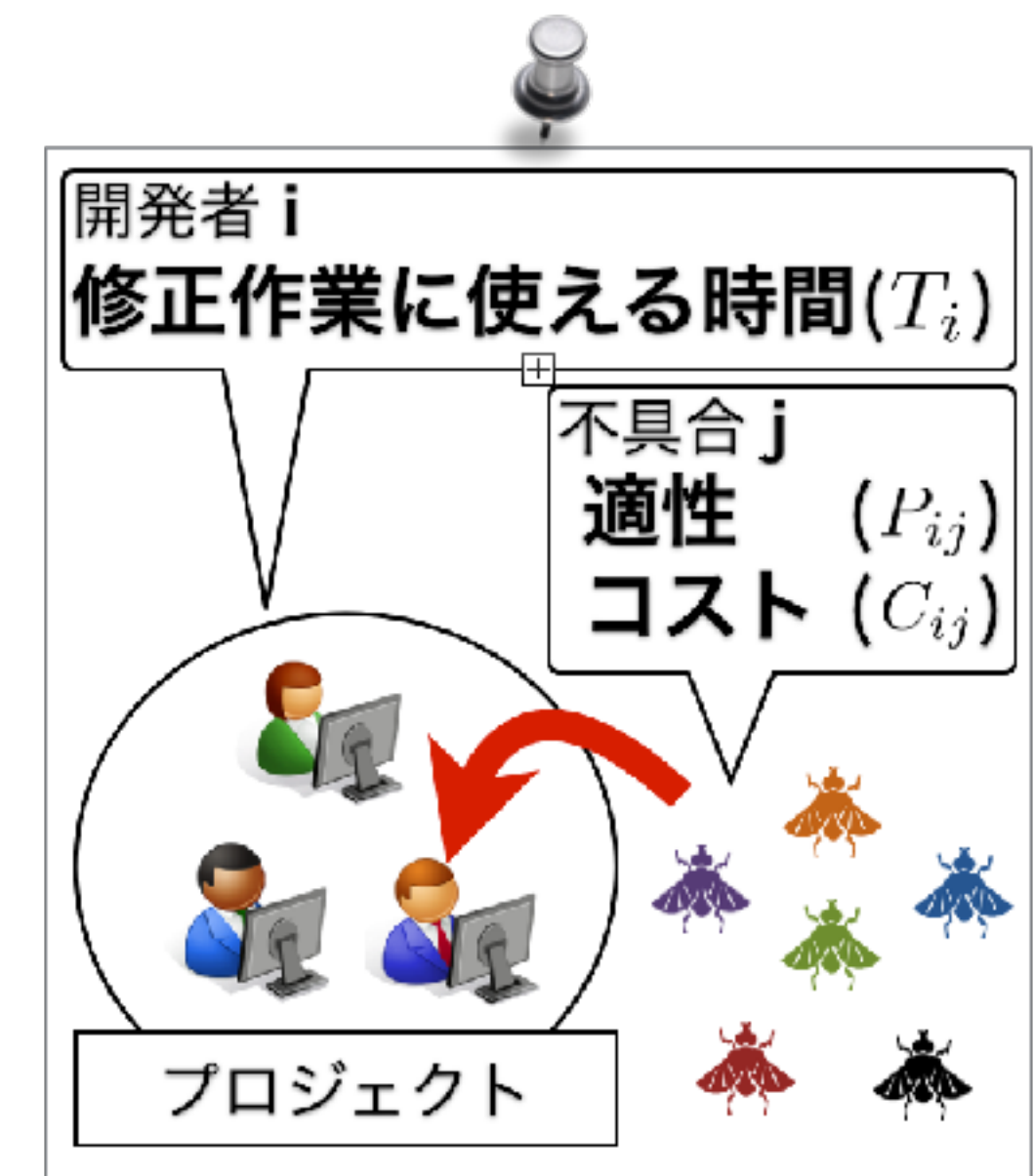
制約条件：

- 各開発者が修正作業に使える時間 (T) は有限

$$\sum_{j=1}^n C_{ij} x_{ij} \leq T_i \quad (i = 1, \dots, m)$$

- 不具合の担当は1人未満

$$\sum_{i=1}^m x_{ij} \leq 1 \quad (j = 1, \dots, n)$$



実験の概要

目的

提案手法がタスクの集中を軽減し、プロジェクト全体で修正活動を効率化できるかを確認する

データセット



GNU Gcc

学習データ：940件
評価データ：250件



Mozilla Firefox

学習データ：1,043件
評価データ：142件

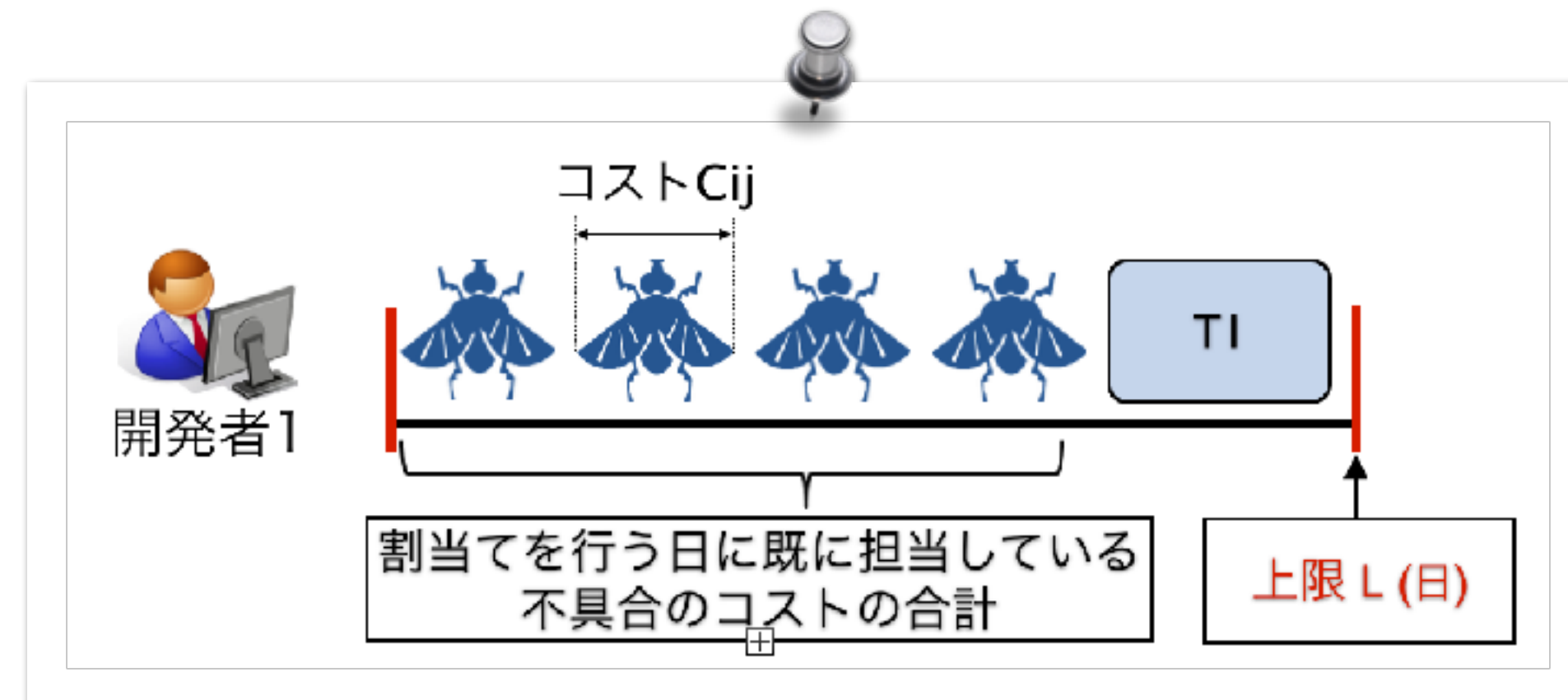


Eclipse Platform

学習データ：783件
評価データ：168件

パラメータ(上限L)

- Gcc : 7日
- Firefox : 14日
- Platform : 9日



比較対象

- CBR(Content-Based Recommendation) [Anvik'06]
SVMを使用し，開発者の適性を重視した手法
- CosTriage [Park'11]
CBRをベースとし，適性が高く，修正時間が短い開発者を割当てる手法

	実験1	実験2	実験3
	割当ての正確さ	修正時間の削減	タスク集中の緩和
CBR	○	×	?
CosTriage	△	○	?
提案手法	?	?	?

実験1: 割当の正確さ

CBRに比べ、割当の精度がどの程度の減少で抑えられるか
(割当てて不具合のコンポーネントを以前に担当したことがあるか)

	既存手法	提案手法
精度	XX%	XX%

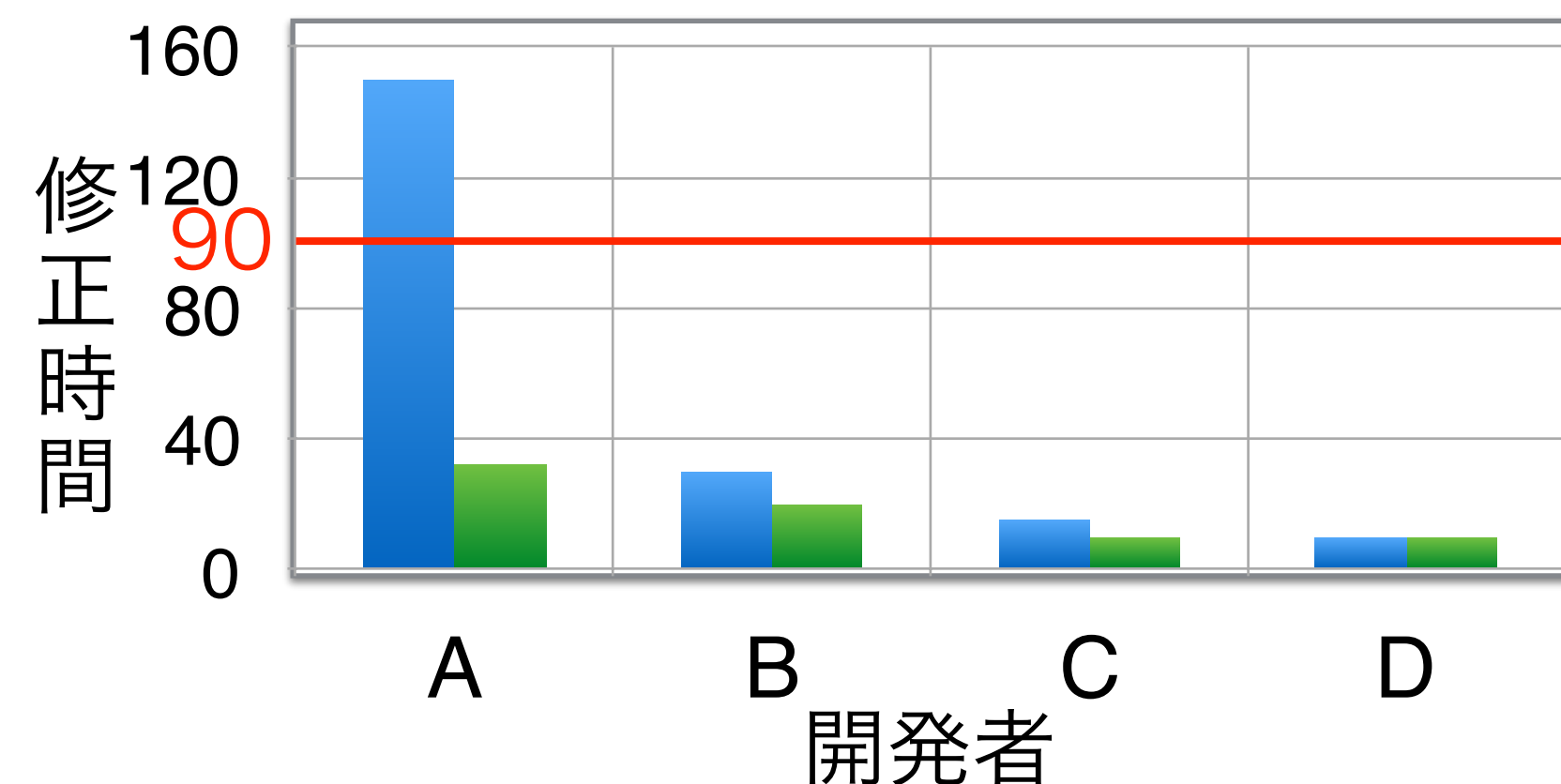
実験2: 修正時間の削減効果

プロジェクト全体の修正時間を削減できるか

	現状の 割当方法	提案手法
修正時間	XXX日	XXX日

実験3: タスク集中の緩和効果

評価データの期間を超えていないか



実験方法

不具合報告の日付に従って不具合の割当てを再現
修正時間はコストを求める方法と同様の方法を用いる



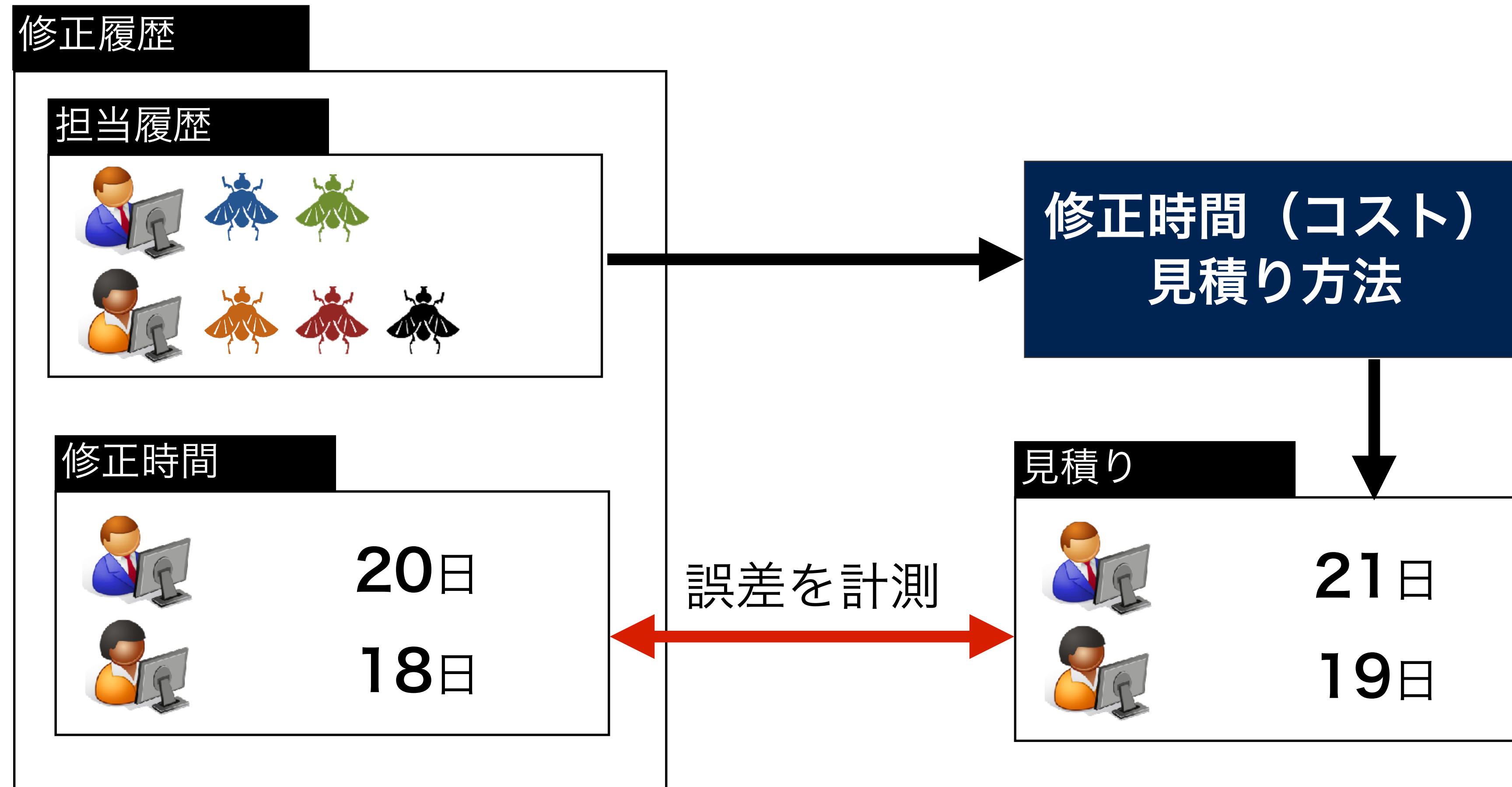
既存手法・提案手法

割当て結果



前実験：修正時間の見積りは正確か

評価データの割当結果とその修正時間を使って，実験における修正時間の見積り精度を確認



前実験：修正時間の見積りは正確か

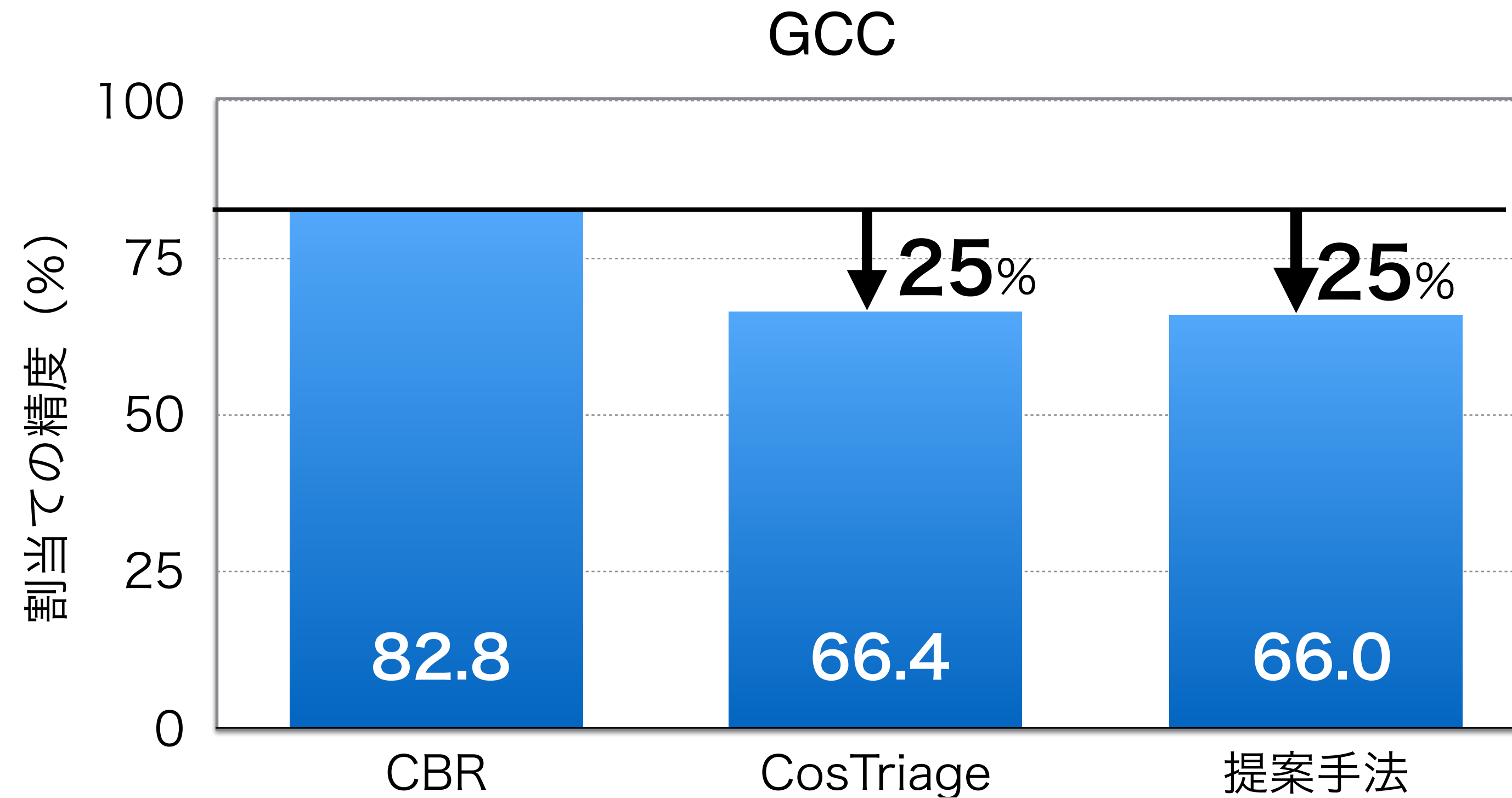
テストデータの割当結果とその修正時間を使って見積もり精度を確認

	実際の 修正時間	修正時間の 見積り	誤差	不具合 一件 あたりの誤差
Gcc	1,089 _日	1,213 _日	124 _日	0.5_日
Firefox	1,702 _日	1,812 _日	110 _日	0.8_日
Platform	1,003 _日	935 _日	-68 _日	0.8_日

実験1：割当ての正確さ

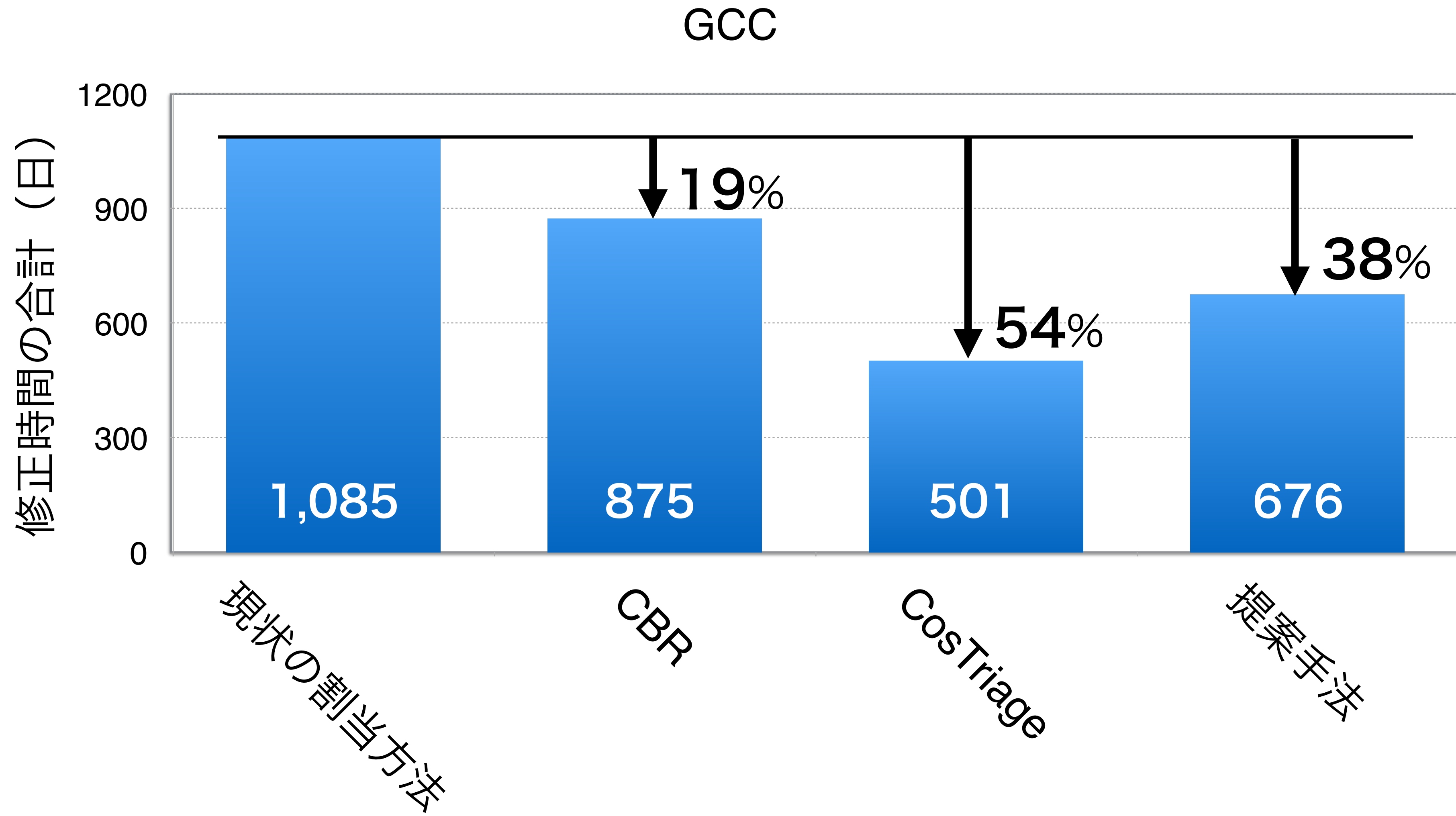
21

既存手法に比べ， 割当精度がどの程度の減少で抑えられるか



実験2：修正時間の削減効果

プロジェクト全体での修正時間を比較

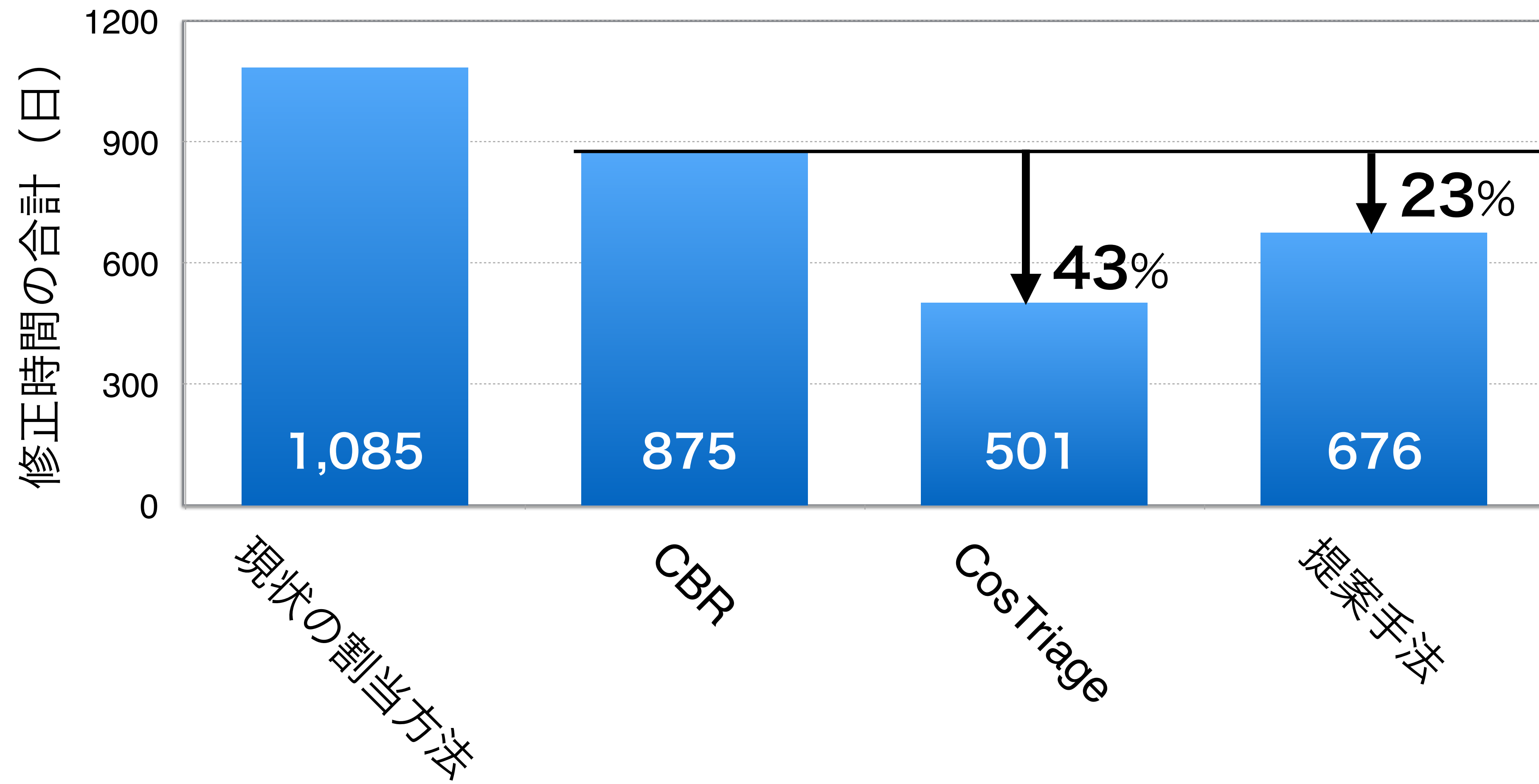


実験2：修正時間の削減効果

23

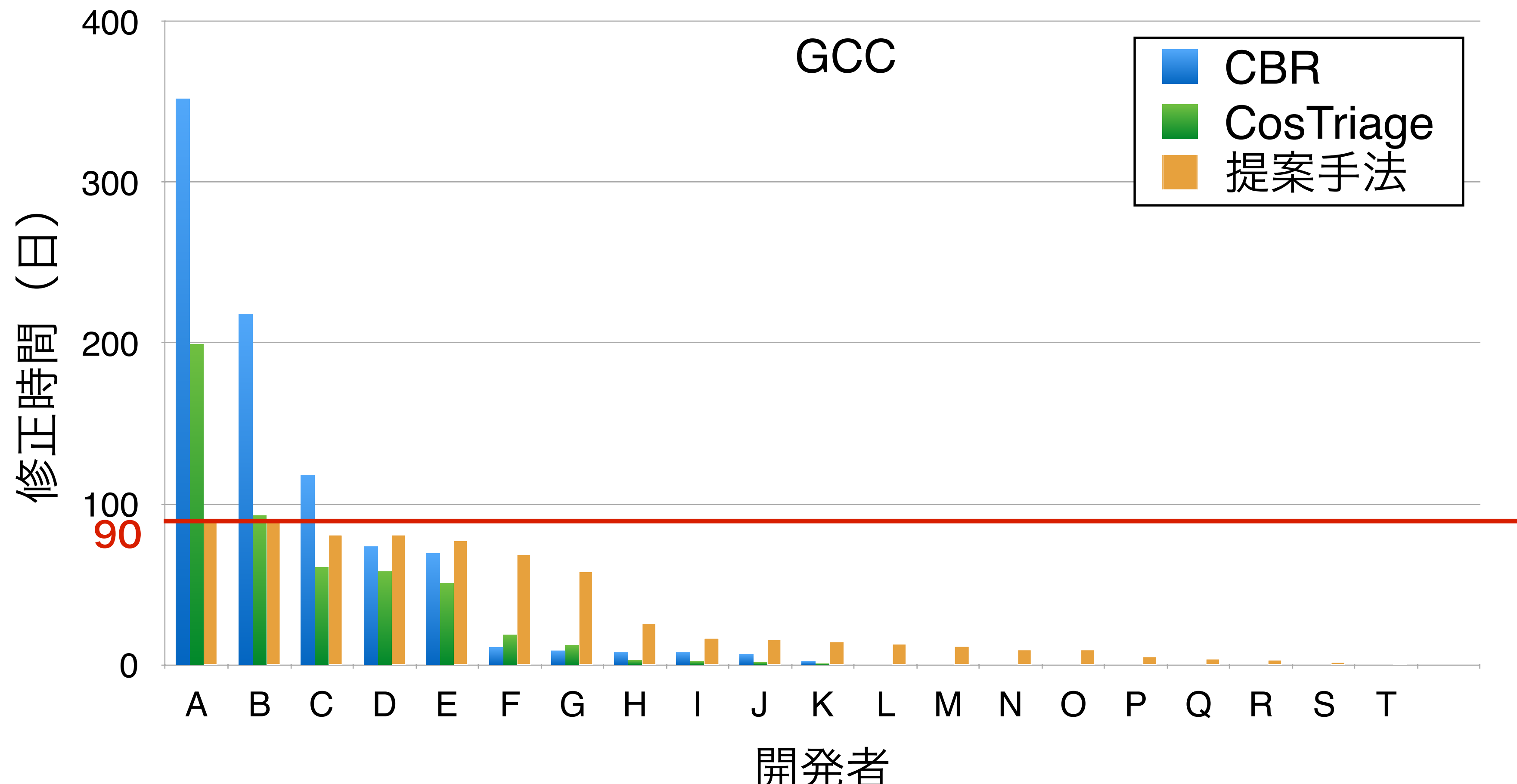
プロジェクト全体での修正時間を比較

GCC



実験3：タスク集中の緩和効果

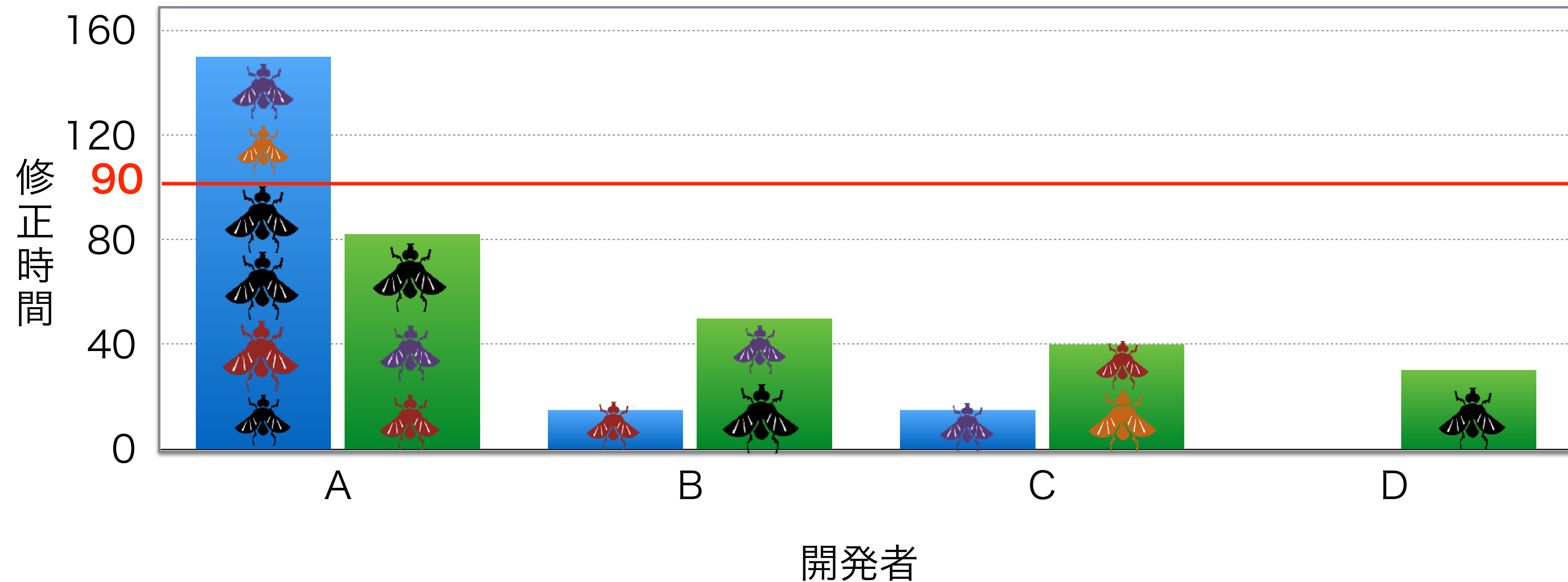
各開発者の修正時間が、評価データの期間より、超えていないかを確認（期間：90日）



考察：タスク集中の緩和効果

リリースまでに修正できた不具合数を確認

割当られた不具合順にタスクが完了したと想定



考察：タスク集中の緩和効果

リリースまでに修正できた不具合数を確認

割当られた不具合順にタスクが完了したと想定

Gcc	CBR	CosTriage	提案手法
不具合数	250件		
完了	108件	166件	226件
未完了	142件	84件	24件

提案手法はリリースが存在するプロジェクトに有効

実験のまとめ

•CBR(Content-Based Recommendation) [Anvik'06]

SVMを使用した手法

•CosTriage [Park'11]

Anvikの手法をベースとして修正時間を考慮した手法

	実験1	実験2	実験3
	割当ての正確さ	修正時間の削減	タスク集中の緩和
CBR	○	×	×
CosTriage	△	○	△
提案手法	△	△	○

リリースまでに修正できる不具合数が増加

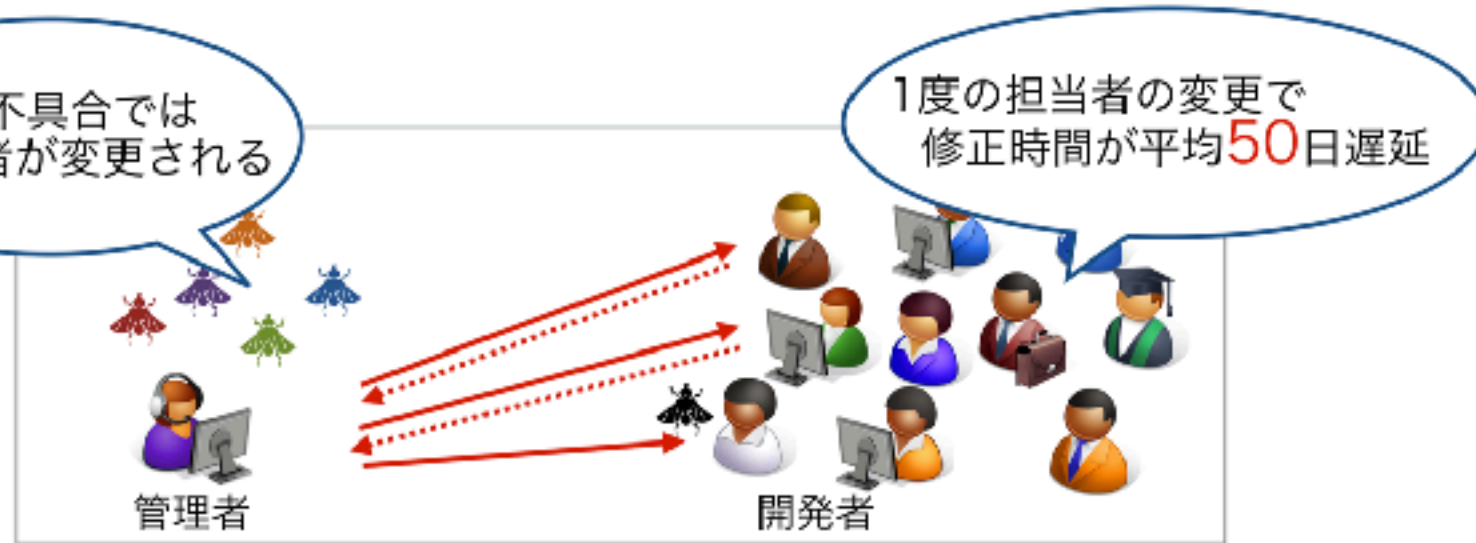
現状のタスク割当て手法と問題点

2

管理者が不具合報告の内容を精査し、担当の開発者を決定

44%の不具合では
担当者が変更される

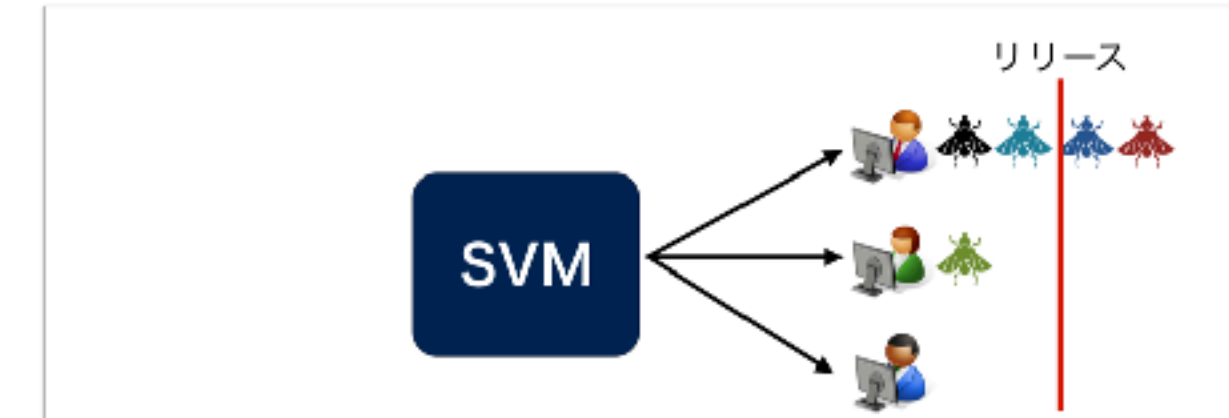
1度の担当者の変更で
修正時間が平均50日遅延



既存手法とその問題点

3

機械学習 (SVM) を用いたタスク割当て



メリット:

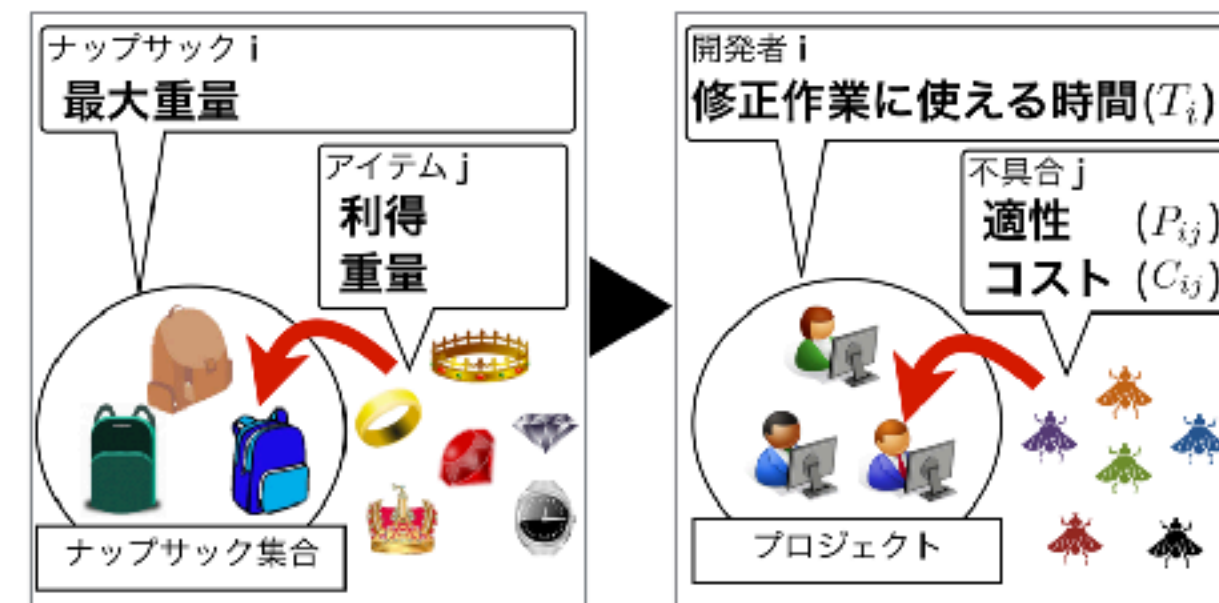
○ 適任の開発者に割当てることができる

デメリット:

✕ 一部の開発者にタスクが集中する傾向

キーアイデア

7



プロジェクト全体で修正効率が最も良くなる
不具合と開発者の組合せを求める

実験のまとめ

26

•CBR(Content-Based Recommendation) [Anvik'06]

SVMを使用した手法

•CosTriage [Park'11]

Anvikの手法をベースとして修正時間を考慮した手法

	実験1	実験2	実験3
	割当ての正確さ	修正時間の削減	タスク集中の緩和
CBR	○	✕	✕
CosTriage	△	○	△
提案手法	△	△	○

リリースまでに修正できる不具合数が増加

今後の課題

- 目的関数の改善（重大な不具合を，より優秀な開発者に割当て）
 - 重要度や優先度の考慮
 - 開発者の能力を数値化

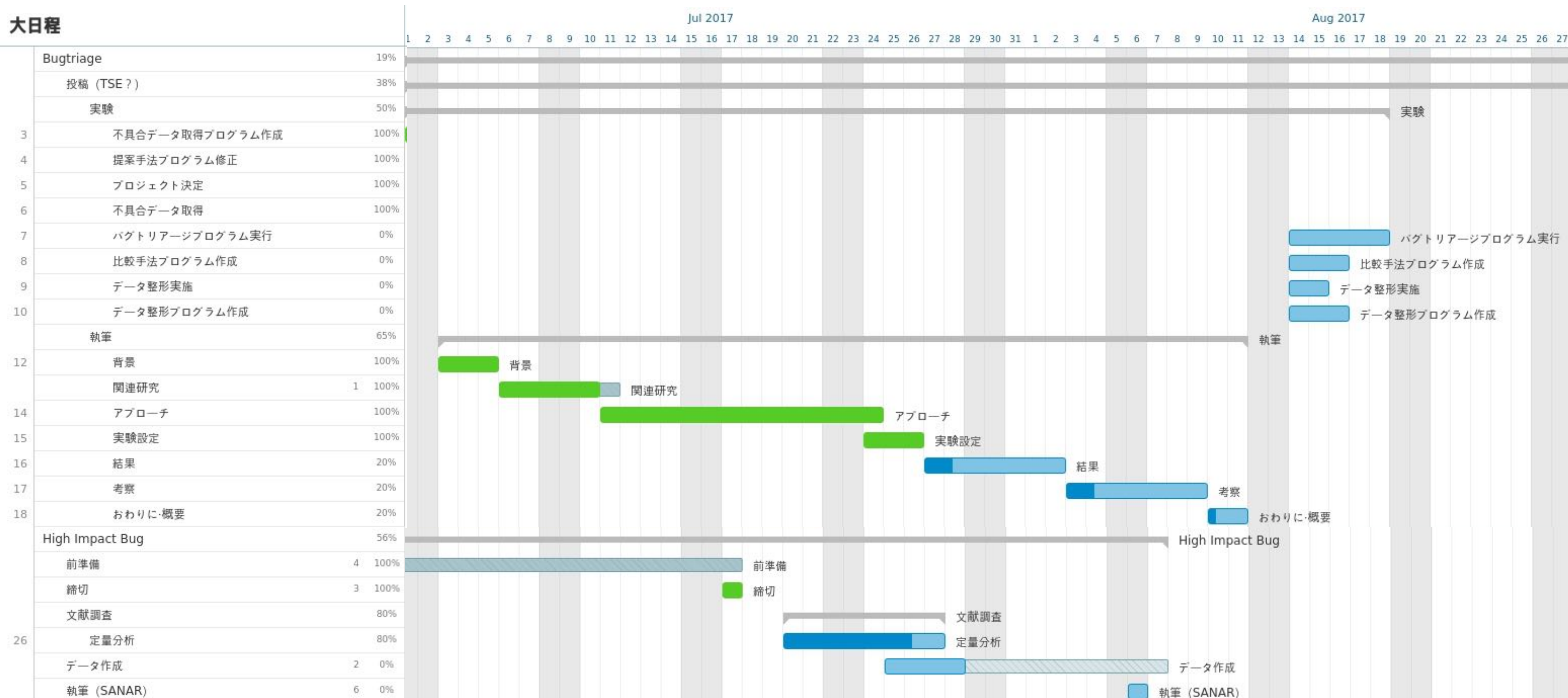
$$\text{Maximize : } \sum_{i=1}^m \sum_{j=1}^n P_{ij} x_{ij}$$

理想は. . .

専門性×開発者の能力×バグの重み

- 再割当が発生する確率を評価
CBRと比べ，割当ての精度が 25% 低下することによる影響
- NP困難である問題に対してどのように対処するか
オーダーの計測，最善の解法アルゴリズムの選択

大日程



執筆で気になっているところ

- 考察：タスク集中の緩和効果はRQに入れるべきでは？
- 考察：プリファレンス・上限の有無比較は不要？

考察：タスク集中効果の要因

本研究で用いた設定（プリファレンスと上限L）が、
どのようにタスクの分散に効果があるかを確認

	上限 無	上限 有
プリファレンス 無	修正時間の短い開発者に 優先的に割当てる	上限を設定した条件下、 修正時間の短い開発者に 優先的に割当てる
プリファレンス 有	プリファレンスが高い 開発者に優先的に割当てる	上限を設定した条件下、 プリファレンスが高い 開発者に優先的に割当てる

考察：タスク集中効果の要因

各モデルが割当てた開発者数

	上限 無	上限 有
プリファレンス 無	13人	18人
プリファレンス 有	11人	20人

既存手法とその問題点

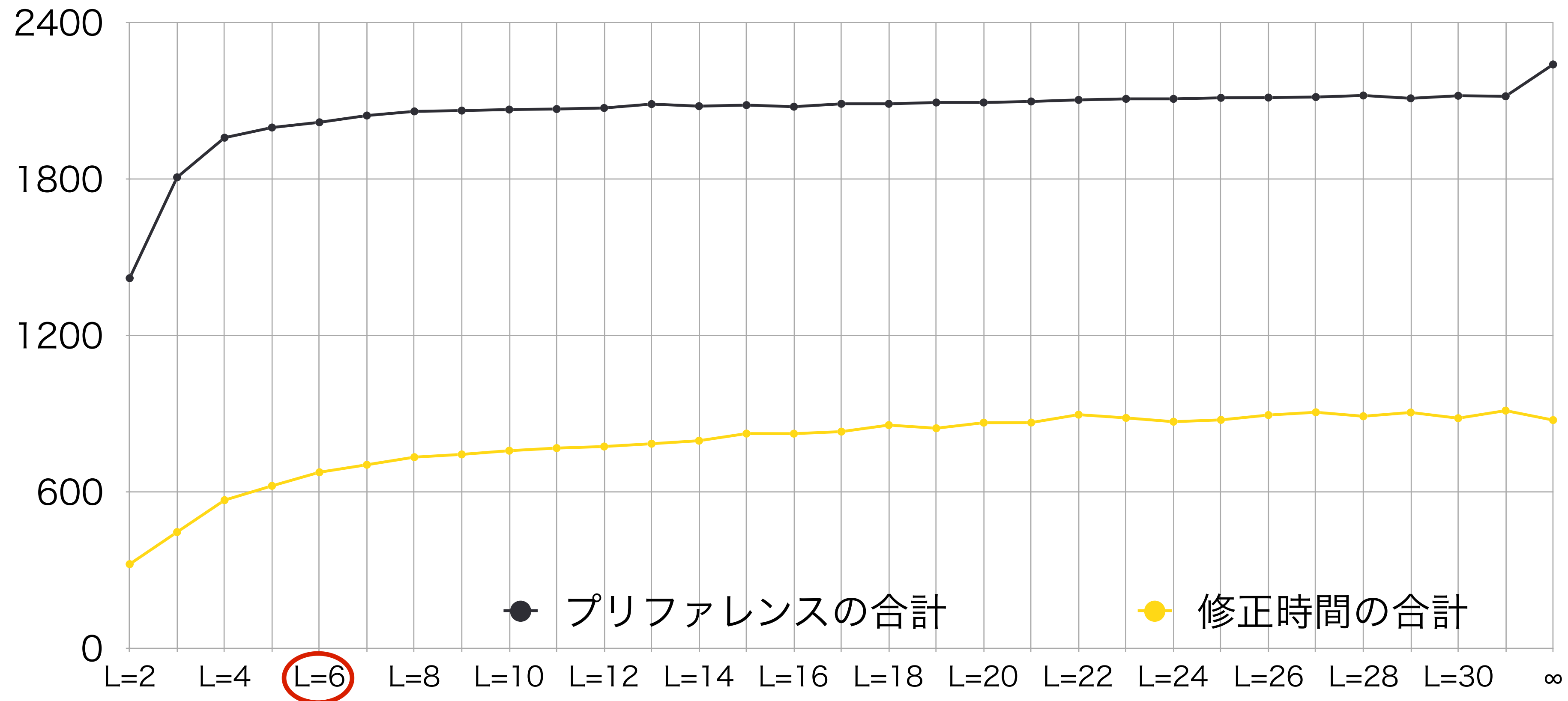
機械学習に基づく不具合の割当 [Anvik'06]



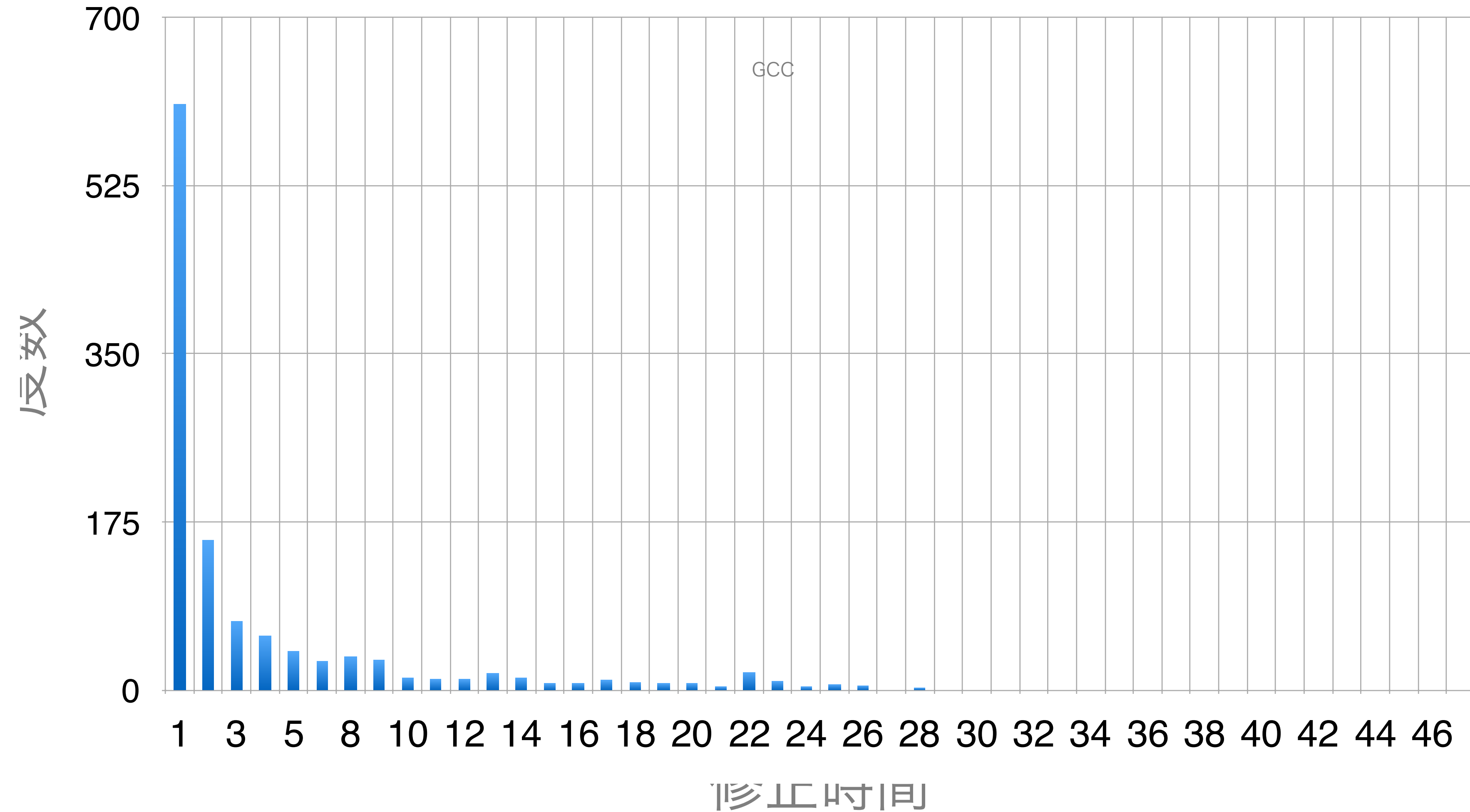
一部の有能な開発者にタスクが集中する傾向

上限Lの変化

- 上限Lの大きさを, 修正時間の合計とプリファレンスの合計がどのように変化するか



修正時間のヒストグラム



Latent Dirichlet Allocation

- 文章内に出てくる単語の共起関係を利用
{ユーザー, アカウント, パスワード, 認証}
{ボタン, クリック, マウス, ドラッグ, ドロップ}
- 複数の文章から共起関係から,
文章がどのような種類の文章かを確率的に求められる
似たトピックの文章は出てくる単語も似ている

本研究での例



A **crash** happens when a **tab** is **dragged** and **dropped** onto a **page**.

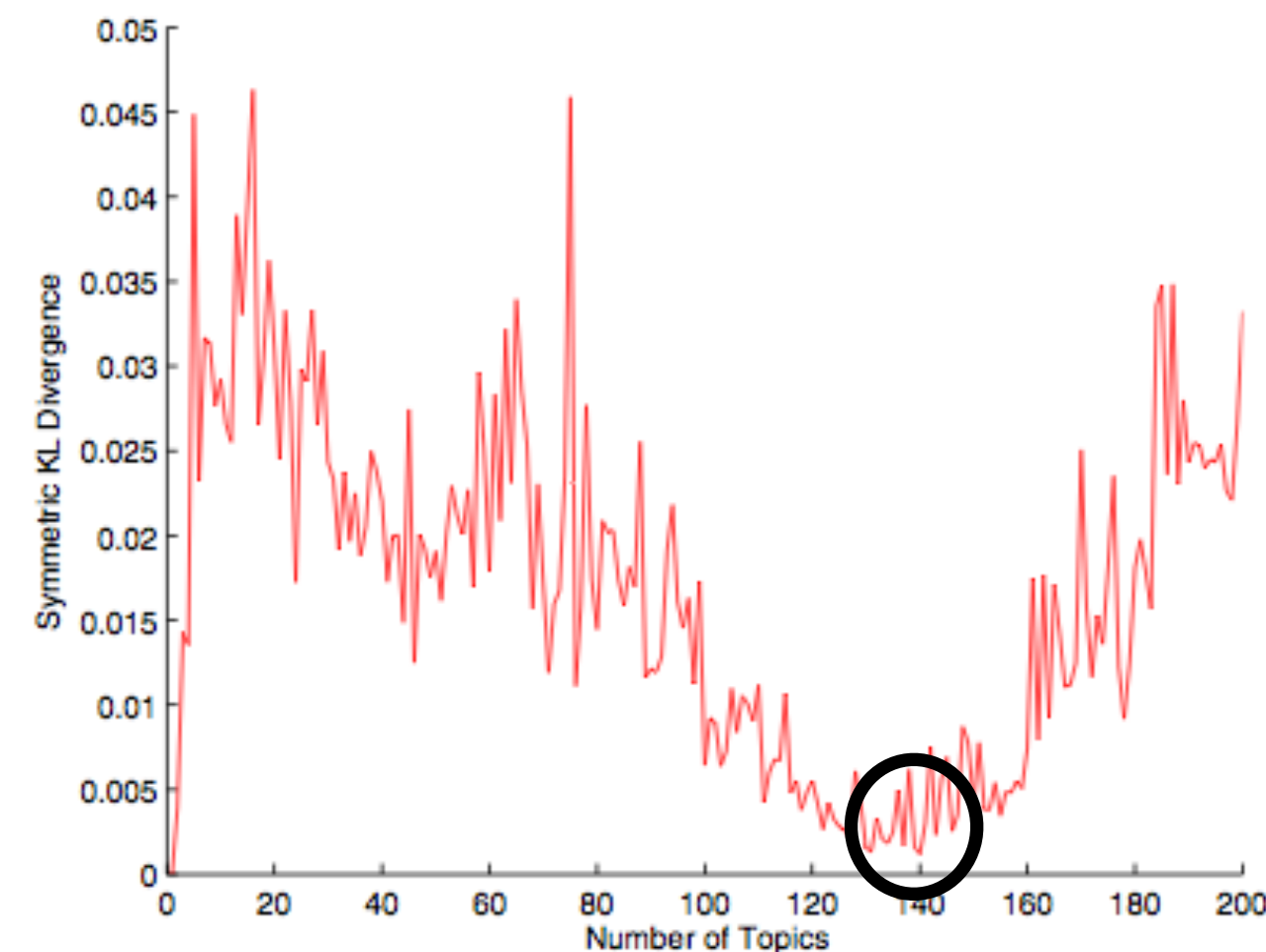
トピック数の決め方 [Arun'10]

- KL Divergence が小さくなるトピック数を用いる
- 2つの確率分布の差異を計る尺度

$$D_{\text{KL}}(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

- LDA適用時に，導出できる2つの行列の差異を測る

(Topic-Word matrix, Document-Topic matrix)



外部発表

- 論文誌

- 柏 祐太郎, 大平 雅雄, 阿萬 裕久, 亀井 靖高, “大規模OSS開発における不具合修正時間の短縮化を目的としたバグトリージ手法," 情報処理学会論文誌, pages 66--75, 2014年2月.

(特選論文)

- 国際会議

- Yutaro Kashiwa, Hayato Yoshiyuki, Yusuke Kukita, and Masao Ohira, "A Pilot Study of Diversity in High Impact Bugs," In Proceedings of 30th International Conference on Software Maintenance and Evolution (ICSME2014), pages 536--540, 2014.

- 国内会議

- 柏 祐太郎, 大平 雅雄, 阿萬 裕久, 亀井 靖高, "大規模OSS開発における不具合修正時間の短縮化を目的としたバグトリージ手法," ソフトウェアエンジニアリングシンポジウム2014論文集, pages 66--75, 2014年8月. **(最優秀論文賞 受賞)**
- 柏 祐太郎, 大平 雅雄, "不具合修正時間の短縮化を目的としたバグトリージ手法の提案," ソフトウェア工学の基礎XX, 日本ソフトウェア科学会 FOSE2013, volume 20, pages 053--058, 2013年11月.
- 柏 祐太郎, 大平 雅雄, "不具合修正タスク量の最適化を目的としたバグトリージ手法の提案," ソフトウェアエンジニアリングシンポジウム2013論文集, pages 1--2, 2013年9月.

(インタラクティブ賞 受賞)