0-1 整数計画法を用いた不具合修正タスクの割当支援

柏 祐太郎[†] 大平 雅雄[†] 阿萬 裕久[*]

本研究の目的は，大規模開発における不具合修正の効率化を支援することである．本研究では， 0-1 整数計画法を不具合修正タスクの割当問題に応用した手法を提案する．本稿では，提案手法について議論するとともに，今後の課題について述べる．

Supporting assignments of bug fixing tasks with 0-1 integer programming

Yutaro Kashiwa[†] Masao Ohira[†] Hirohisa Aman[*]

The goal of our study is to support efficient bug fixing in a large-scale software development. We propose a method for task assignments in fixing bugs by applying a 0-1 integer programming technique to the task assignment problem. This paper describes how we use the 0-1 integer programming for the task assignments and discuss our future work.

1. はじめに

近年，大規模 Open Source Software (OSS) 開発プロジェクトでは，不具合修正の長期化が問題となっている．主な原因は，不具合修正タスクの割当てがうまくいかず，不具合担当者の変更，すなわち再割当て (reassignment) が頻発することと指摘されている[1][2]．実際，Eclipseプロジェクトの44%の不具合は，再割当ての結果，2 人以上の担当者が修正している[1]．

リポジトリマイニング（MSR）の分野では，再割当ての回数を減らすための推薦方法[1][2]や，自然言語処理を用いて各不具合修正タスクに適任の開発者を推薦する方法[3]など，不具合修正タスクの再割当てを解決するための数多くの手法が提案されている．しかしながら，既存アプローチの多くは，有能な特定の開発者に不具合修正タスクが集中する傾向にある（一人で大量に不具合修正タスクを抱える）ため，プロジェクト全体としては，必ずしも不具合修正の長期化を解決できる訳ではない．

2. 目的とアプローチ

本研究の目的は，開発者が一定期間に担当可能な不具合修正タスクに上限があることを前提として，プロジェクト全体として不具合修正の効率を改善するためのタスク割当手法を構築することである．本研究では，不具合修正タスクをプロジェクト全体として効率的に割り当てる方法として，0-1 整数計画問題[4]を応用した手法を提案する．

2.1. 0-1 整数計画問題

0-1 整数計画問題とは，何かを選ぶか選ばないかという決定問題を定式化する際に用いられ，制約条件の下，全てのアイテム集合から最善の部分アイテム集合を見つけ出す問題である．例えば，それぞれ異なる重さと価値を持つ品物を，一定の容量の袋に価値が最大となるような組み合わせで詰め込むような問題，いわゆるナップサック問題である．

2.2. 不具合修正タスクの割当への応用

本研究では，開発者D_i $(i = 1,2,3, \dots, m)$ と不具合B_j $(j = 1,2,3, \dots, n)$ が存在する環境下で，どの開発者がどの不具合を担当すればプロジェクト全体として最も効率的に不具合修正が行えるかについての組み合わせ問題を，0-1 整数計画問題[4]として解く．

不具合修正タスクの割当問題に対して0-1整数計画問題を応用するために，本研究では目的変数，目的関数および制約条件を以下のように定義する．図1は 0-1 整数計画法の適用イメージである．

目的変数：開発者D_iに対し，不具合B_jを割当てるか割当てないかを 0-1 変数として表す．

x_{ij}={0,1}

$x_{ij} = 1$ は開発者D_iに不具合B_jを割当てることを意味し，$x_{ij} = 0$ は割当てないことを意味する．

†和歌山大学, Wakayama University
*愛媛大学, Ehime University

©2013 Information Processing Society of Japan

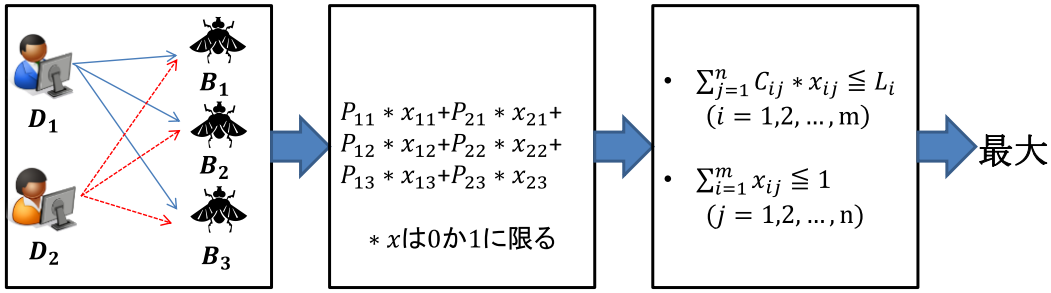

図 1　0-1 整数計画法の適用イメージ

目的関数:どの開発者がどの不具合を担当すればプリファレンスの総和が最大となるかについての組み合わせを求める.

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

ここで, プリファレンス (P_{ij}) とは, 開発者D_i が不具合B_jの修正にどのくらい向いているかを示す尺度とする. 本研究では, 不具合の対象コンポーネントと不具合解決の優先度を用いて不具合を分類した後, 各カテゴリでの過去の全修正不具合数に対する開発者D_i の修正不具合数の比をプリファレンスと定義する.

$$P_{ij} = \frac{\text{不具合}B_j\text{が属するカテゴリにおける開発者}D_i\text{の過去の修正数}}{\text{不具合}B_j\text{が属するカテゴリにおける全開発者の過去の修正数}} * 100$$

制約条件:本研究では, 以下の 2 つを制約条件とする.

1. 各開発者が一定期間内に修正できる不具合数は有限であること.
   $\sum_{j=1}^{n} C_{ij} * x_{ij} \leqq L_i \quad (i = 1,2, \ldots, m)$
2. 各不具合の担当者は1人以下であること.
   $\sum_{i=1}^{m} x_{ij} \leqq 1 \quad (j = 1,2, \ldots, n)$

ここで, コスト (C_{ij}) は, 開発者D_i が不具合B_jの属するカテゴリの不具合を修正するときの平均修正時間(中央値)とする. また, 上限 (L_i) は, 0-1 整数計画法を用いる際の上限設定であり, 開発者D_iが一定期間内に不具合修正作業をあと何日行えるかを示すものである. L_i は, 不具合修正タスクの割当てを行う際に, 開発者D_i がすでに担当している不具合の数を基に算出される.

本上限設定により, 割当時点ですでに数多くの不具合修正タスクを担当している開発者の新規割当数(割当数の上限)は低く設定されるため, 特定の開発者にタスクが極端に集中するのを防ぐ効果を期待できる.

3. まとめと今後の課題

本研究では, 不具合修正タスクが特定の開発者に集中するという先行研究の問題点に着目し, 開発者に多大な負担がかかることを緩和しつつ効率的な不具合修正タスクの割当を実現するために, 0-1 整数計画法を応用する手法を提案した.

今後は複数の大規模 OSS プロジェクトに本手法を適用し, 不具合修正の効率改善の効果を確認する. また, 0-1 整数計画法との比較として貪欲法を用いた実験を行い, 本手法の有意性を示す. また, 上限L_i の大きさを変更し, 最適のL_i を見つけるためのシミュレーション実験を行う.

謝辞

本研究の一部は, 文部科学省科学研究補助金(基盤(B): 23300009)および(基盤(C): 24500041)による助成を受けた.

参考文献

[1] Gaeul Jeong, Sunghun Kim and Thomas Zimmerman, Improving Bug Triage with Bug Tossing Graph, In Proceedings of The 7th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE'09), pp. 111-120, 2009.

[2] Pamela Bhattacharya and Iulian Neamtiu, Fine-grained Incremental Learning and Multi-feature Tossing Graphs to Improve Bug Triaging, In Proceedings of 26th IEEE International Conference on Software Maintenance (ICSM'10), pp. 1-10, 2010.

[3] John Anvic, Lyndon Hiew and Gail C. Murphy, Who Should Fix This Bug?, In proceedings of 28th International Conference on Software Engineering (ICSE'06), pp. 361-370, 2006.

[4] 福島雅夫, 新版数理計画入門, 朝倉書店, 2011.

©2013 Information Processing Society of Japan