配付資料のテンプレート

名大 太郎

2018 年 1 月 8 日

Abstract. Write an abstract of five to ten lines in English even if you write the main body in Japanese. Abstract abstract abstract abstract abstract abstract abstract abstract abstract. Abstract abstract abstract abstract abstract abstract abstract abstract abstract. Abstract abstract abstract abstract abstract abstract abstract abstract abstract. Abstract abstract abstract abstract abstract abstract abstract abstract abstract. Abstract abstract abstract abstract abstract abstract abstract abstract abstract.

1 はじめに

これは配付資料のテンプレートです．配付資料を作る前に，書き物に関する基本的な注意点を書いたページ[*1]に書いたことをよく読んでください．このページには加筆，訂正することがときどきあるので，資料を作るたびに見直してください．

発表のたびに配付資料を書く（あるいは前回の資料に加筆修正する）ことで，論文のための文章を書くことや LaTeX を使うことに慣れるとともに，卒論・修論につながる文章を蓄積していってください．

このテンプレートはあくまでも参考ですので，フォントサイズや行間等のスタイルを自由に変更してかまいません．

「はじめに」の節ではこの資料で何を書くのかを手短に説明してください．どのような問題を対象にしてどのような手法を提案し，どのような結果が得られたのかを書くなどです．また，研究背景についても関連研究の文献を挙げつつ述べましょう．文献リストの書き方の例を挙げておきます．[1] は本，[4, 5] は論文誌の論文，[2] は国際会議の予稿集の論文，[3] は論文を集めた本やハンドブックに掲載された論文の例です．

必要な情報を順序よく（つまり後ろを見ないと分からないことが出て来たりしないように），明確に（曖昧さなく厳密に），コンパクトに（冗長な表現を無くして手短に）書くよう心がけましょう．

構成は研究テーマや書きたいことによって様々ですが，例えばある問題に対するアルゴリズムを提案して，その効果を計算実験によって検証するようなテーマであれば，はじめにの節ののち，問題の説明，提案手法の説明，計算実験の紹介，まとめという構成がしばしば用いられます．以下の節ではそのような説明を書く際によく利用する書式や書くべきことの例をいくつか示しておきます．

2 問題

問題の定義を書くときには，何が与えられて（i.e., 入力）何を決めることが求められているのか（i.e., 決定変数），解が満たすべき条件（i.e., 制約条件），および解の良さを判断する基準（i.e., 目的関数）を明確にしてください．整数計画問題として自然な定式化が可能な場合はそのような定式化を与えるとよいと思います．フォーマットの一例を挙げておきます:

$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^{n} p_j x_j & \\ \text{subject to} \quad & \sum_{j=1}^{n} w_{ij} x_j \le c_i, & \forall i \in I \\ & x_j \in \{0,1\}, & \forall j \in J. \end{aligned}$$

一方，整数計画問題に書こうとすると複雑になって返って分かりにくい場合は，その必要はありません．例えば，順列や割当を表す写像を用いる定式化もしばしば有効です．例えば一般化割当問題と呼ばれる

[*1] http://www.co.mi.i.nagoya-u.ac.jp/~yagiura/writing/writing.html

問題は，整数計画問題として

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} & \\
\text{subject to} \quad & \sum_{j=1}^{n} a_{ij}x_{ij} \le b_i, & \forall i \in I \\
& \sum_{i=1}^{m} x_{ij} = 1, & \forall j \in J \\
& x_{ij} \in \{0,1\}, & \forall i \in I,\ \forall j \in J
\end{aligned}
$$

のように定式化できますが，割当 $\pi : J \to I$ ($\pi(j) = i$ は j を i に割り当てることを表す) を用いて

$$
\begin{aligned}
\text{minimize} \quad & \sum_{j=1}^{n} c_{\pi(j),j} & \\
\text{subject to} \quad & \sum_{j=1}^{n} a_{\pi(j),j} \le b_i, & \forall i \in I
\end{aligned}
$$

と少しコンパクトに書くこともできます．この問題の場合はどちらも簡潔で分かりやすいので，議論に便利な方を選べば良いと思います．どちらの定式化でもその後の説明を書く上で大差ないのであれば，短い方を選ぶのが良いと思います．

関連文献等を調査して，問題に対して分かっていることも書きましょう．研究内容によって書くべきことは変わりますが，例えば NP 困難性などの計算の複雑さ，近似精度保証の上界と下界，多項式時間で解ける特殊ケースとそのような場合の計算量などの理論的に解明されている性質や，どのような問題例が代表的なベンチマーク問題例として利用されていて，どの程度の規模までがどの程度の時間で厳密に解かれているか，メタ戦略のような解法の比較の対象としてはどの程度の規模が対象になっているかなど．

3 提案手法

提案手法を書く際には，

1. まず手法の大まかなアイデアや全体像を言葉で簡潔に説明し，
2. 次に手法の構成要素のおのおのの詳細を述べ，
3. 最後にそれらの構成要素をどのように組み合わせて全体の枠組みが構成されているのかを示す

というような順序で書くと分かりやすいと思います．その際，各構成要素を，サブルーチンのように入力として何を受け取って何を返すのかを記述した手続きとして名前をつけてまとめておき，最後に全体の枠組みを示す際にそれらを利用してアルゴリズムを記述すると書きやすいと思います．

このような手続きを疑似コードとしてまとめるのに `algorithmic` が便利です（`algorithmicx` も便利のようです）．Algorithm 1 に例を示します．

Algorithm 1 Ford-Fulkerson

Input: グラフ $G = (V, E)$，始点 $s \in V$, 終点 $t \in V$, および各辺 $e \in E$ の容量 u_e.

Output: s から t への最大フロー.

1: **for** $e = 1$ to $|E|$ **do**
2: 　$x_e := 0$ とする.
3: **end for**
4: 残余ネットワーク G_x を作成する.
5: **while** G_x にフロー追加路が存在する **do**
6: 　フロー追加路に沿って x にフローを追加する.
7: 　残余ネットワーク G_x を更新する.
8: **end while**
9: フロー x を出力して終了.

なお，手続きの入出力を表す「`\REQUIRE`」と「`\ENSURE`」を使うと，通常はそれぞれ「**Require:**」と「**Ensure:**」のように表示されますが，これらを上の例のように「**Input:**」と「**Output:**」に変更するための記述が本 LaTeX ファイルのプリアンブル[*2]にあります．

手続きをまとめるのに以下のようなスタイルを使うこともあります．なお，Step 番号を `description` 環境で手動で書いても同様のスタイルを実現できますが，Step を追加したり削除したりしたときに番号の相互参照で失敗しないよう，自動的に番号を振る方法の方がよいと思います．以下の例では `enumerate` 環境を利用していますが，プリアンブル

[*2] `\documentclass` と `\begin{document}` の間の部分.

に\usepackage{enumerate}が必要です.

Algorithm FORD-FULKERSON

Input: グラフ $G = (V, E)$ . . .（略）

Output: s から t への最大フロー.

Step 1. すべての $e \in E$ に対し $x_e := 0$ とする.

Step 2. 残余ネットワーク G_x を作成する.

Step 3. G_x にフロー追加路が存在しなければ，フロー x を出力して終了.

Step 4. フロー追加路に沿って x にフローを追加.

Step 5. G_x を更新したのち Step 3 に戻る.

どのようなスタイルを使う方が書きやすいか，分かりやすいかは，アルゴリズムによっても変わりますし，好みもあると思いますが，アルゴリズムの反復構造は algorithmic の方が分かりやすいように思います.

疑似コードを読まなくてもアルゴリズムのアイデアが大体分かるように本文の文章中に説明を書きましょう．疑似コードはアルゴリズムを正確に記述するためのものなので，複雑になりがちです．従って，よほど内容に興味がある読者でなければ疑似コードまで読みたいとは思いません．読者が疑似コードを読まなくてもアイデアの概要が本文から分かるように書いてなければ，とても読みづらい論文になってしまいます.

4 実験結果

実験結果を示す際には，計算環境（実験に用いた計算機の CPU やメモリー, 実装に用いた言語）を明記しましょう．計算結果を表示するのに用いる図や表を表示する例を図 1 と表 1 に示しておきます．また，2 段組みの原稿において幅の広い表を表示する例を表 2 に示しておきます（図も同様に\begin{figure*}のようにすればよい）.

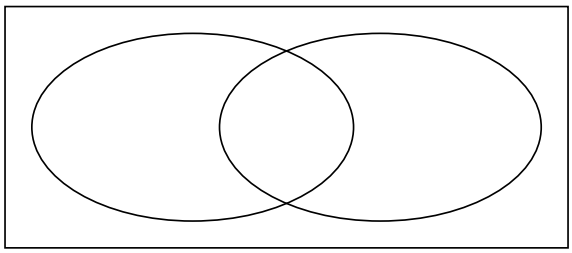

図 1 図の表示例

表 1 表の表示例

問題例	最良値	計算時間（秒）
c05100	123	10.1
c10100	456	15.2
c20100	789	20.3

5 まとめ

まとめは著者から読者への締めくくりの言葉です．すなわち，「この論文のポイントは結局何だったのか」を端的に読者に示す大切な部分です．必ず書きましょう[*3]．結局何をしてどうなったのかということを最後にもう一度手短にまとめて述べます．この節で新しいこと（つまりこれまでの節で書いて来なかったこと）を書かないのが原則です.

通常の論文等ではこのように得られた成果をまとめた結論を書いて終わるのですが，研究の進捗状況を報告する普段の研究会の発表では，結論を書くことは難しいかもしれません．また，既に一定の成果が得られている場合でも，卒論・修論の締切が差し迫った時期を除き，卒論・修論に向けてさらに研究を進める予定であると思います．そのような場合には，この節のタイトルを例えば「まとめと今後の研究計画」などとして，まず現在までに得られている成果をまとめたのち，今後の研究計画を簡潔に書いてください.

[*3] レター（ページ数の少ない速報的な論文（e.g., Operations Research Letters, Information Processing Letters））などの短いものではまとめの節を書かないように指示されることもあり，そのような場合を除く.

表 2　2 段組みスタイルにおいて幅の広い表を表示する例

	既存手法		提案手法	
問題例	最良値	計算時間（秒）	最良値	計算時間（秒）
c05100	123	10.1	111	10.0
c10100	456	15.2	432	15.0
c20100	789	20.3	765	20.0

参考文献

[1] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, New York, 1979.

[2] S. Imahori, M. Yagiura and T. Ibaraki, Variable neighborhood search for the rectangle packing problem, *Proceedings of the 6th Metaheuristics International Conference* (MIC), Vienna, Austria, August 22–26, 2005, pp. 532–537.

[3] D.S. Johnson and L.A. McGeoch, The traveling salesman problem: a case study, in: E.H.L. Aarts and J.K. Lenstra (eds.), *Local Search in Combinatorial Optimization*, John Wiley & Sons, Chichester, 1997, pp. 215–310.

[4] 真野洋平，橋本英樹，柳浦睦憲，学生実験のスケジューリングシステムの構築，オペレーションズ・リサーチ，58 (2013) 524–532.

[5] M. Yagiura, T. Ibaraki and F. Glover, An ejection chain approach for the generalized assignment problem, *INFORMS Journal on Computing*, 16 (2004) 133–151.