

- 1. はじめに
- 2. 自動献立作成の概要
- 4. 提案手法
- 5. おさらい
- 5. 数値実験
- 6. おわりに
- 5. おさらい

ブラウザベースの献立作成システムの 改善による実用化

水上 和秀 (Kazuhide Mizukani)
u355020@st.pu-toyama.ac.jp

富山県立大学 工学部 電子情報工学専攻

December 1, 2023

1.1 研究の背景

2/20

背景

近年、生活習慣病を患う人々が増加している。生活習慣病とは「食習慣、運動習慣、休養、喫煙、飲酒、ストレスなどの生活習慣を原因として発症する疾患の総称」のことであり、深刻な疾患に深く関与している。

生活習慣病を患った場合、食生活を見直すことで改善することができる。しかし栄養バランスの取れた献立を作成するには、メニューの組み合わせや栄養価の計算を考慮する必要がある、献立を考えることは面倒と考える人は少なくない



図1 生活習慣病を起因とする疾患

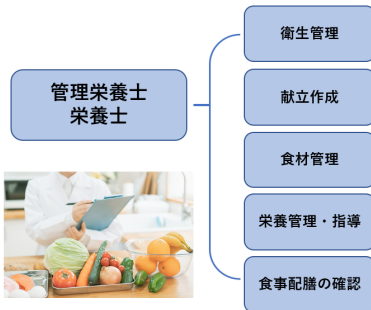


図2 栄養士の主な業務内容

1. はじめに
2. 自動献立作成の概要
4. 提案手法
5. おさらい
5. 数値実験
5. おわりに
5. おさらい

1.2 研究の目的

3/20

目的

献立作成による負担を減らすために、様々な推薦方法が研究されているが、ユーザの嗜好を考慮する機能を持たないため、ユーザに最適なレシピを推薦することは難しいと考えられる。そこで摂るべき栄養素やカロリーが満たされた献立作成をコンピュータによって自動的に行うシステムを提案する。
また、

- 1. はじめに
- 2. 自動献立作成の概要
- 4. 提案手法
- 5. おさらい
- 5. 数値実験
- 6. おわりに
- 5. おさらい

Web 上のレシピデータを活用

4/20

Web 上のレシピデータを活用

システムに使用するレシピとしてレシピサイト「ボブとアンジー」「eatsmart」「おいしい健康」から、料理レシピデータ（必要材料、摂取栄養量、カロリーなど）をスクレイピングし、参照する。食品価格動向を調査しているサイト「小売物価統計調査による価格調査」から様々な食品とその価格データをスクレイピングする。次に、料理レシピデータの食材と食材価格データの食材を照らし合わせて食材コストを計算する。



図7 レシピサイト・ボブとアンジーにおける料理レシピ情報

小売物価統計調査による価格推移

HOME / キャベツの販売価格・価格推移 / 全国

全国のキャベツ 1 kg
価格推移 / 過去84ヵ月



WEBスクレイピング

データ抽出・出力

全国のスーパーで売られているキャベツ 1 kg 単位の平均は124円。
2015年1月～2021年12月（過去84ヵ月）の期間で全国のキャベツが最も高かった最高値は2018年2月で354円、最も低かった最低値は2020年2月で118円となっています。
全国エリアでキャベツの最高値（2018年2月）と最低値（2020年2月）との価格差は236.4321円となっています。
キャベツ 1 kg の2015年1月～2021年12月の価格推移とグラフは下記をご覧ください。
出典元：総務省統計局 小売物価統計調査（2021年12月）
関連新聞：キャベツのふくらみと価格（ヨミウランキン）

図8 食品価格推移調査サイトの例

スクレイピングする主なデータ

- 料理レシピ名
- 調理時間
- 摂取カロリー
- 摂取栄養量
- 必要食材名
- 必要食材量
- 作り方
- 画像URL
- 食材価格
- 販売単位
- 食材名



食材コスト
の計算

図9 Webデータ活用の流れ

献立作成システムは、決められた制約条件の中で、目的関数を最大または最小となる組み合わせの解を探索する、組み合わせ最適化問題として捉えられる。献立作成における制約条件として、栄養素を最低でどれだけとるか、カロリーをどのくらい制限するか、などが挙げられる。また、目的関数として、調理時間と調理コストの最小化が挙げられる

献立作成における 目的関数、制約条件の例

目的関数の例

- ・ 調理時間の最小化
- ・ 個人の嗜好の最大化
- ・ 食材ロスの最小化
- ・ 食材コストの最小化

制約条件の例

- ・ 特定の栄養素量の制限
- ・ 摂取カロリーの制限
- ・ 献立を作成する日数
- ・ 調理工程の制限

献立作成における研究例

- ・ 必要な摂取栄養量をファジィ数で表す、ファジィ数理計画法を用いた献立作成。
- ・ ユーザとの対話型処理によって献立を作成する。
- ・ ユーザの献立の雰囲気によって献立を作成する。



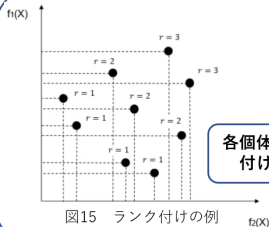
図7 ナップサック問題の例

多目的最適化問題を解く手法として、NSGA-IIを用いる。これは、遺伝的アルゴリズムを多目的最適化問題に拡張したものであり、非優越ソート、混雑度トーナメント選択といった特徴を持つ。

NSGA-IIの特徴

- ・非優越ソート
- ・混雑度トーナメント選択

非優越ソート



混雑度トーナメント選択

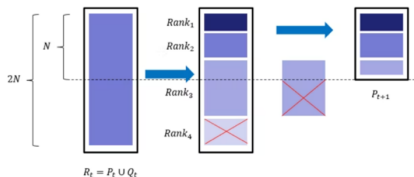


図14 NSGA-IIのアルゴリズム

混雑度トーナメント選択

$$\text{混雑距離} : CD(x^{(i)}) = \frac{1}{k} \sum_{j=1}^k |\tilde{f}_j(x^{i+1}) - \tilde{f}_j(x^{i-1})|$$

- ・個体 i のランクが個体 j のランクよりも優れている。
- ・個体 i と個体 j はともに同じランクであり、 i の混雑距離が j よりも優れている。

調理時間とコストを最小化するパレート最適な献立

7/20

献立に含まれる料理の調理時間の最小化と、料理の食材コストの最小化を目的関数とし、摂取カロリーや摂取栄養量などを制約条件として多目的最適化を行い、パレート最適な献立を出力する。

定式化

$$\begin{aligned}
 &\text{minimize} && \sum_{k=1}^{3D} \sum_{i=1}^R r_{ki} T_i \\
 &\text{minimize} && \sum_{k=1}^{3D} \sum_{i=1}^R r_{ki} G_i \\
 &\text{subject to} && F_l^L \leq \sum_i r_{ki} f_{il} \leq F_l^H \quad (\forall k, \forall l) \\
 & && B^L \leq \sum_i r_{ki} C_i \leq B^H \quad (\forall k) \\
 & && \sum_i r_{ki} T_i \leq \tau_1 \quad (k \% 3 = 1) \\
 & && \sum_i r_{ki} T_i \leq \tau_2 \quad (k \% 3 = 2) \\
 & && \sum_i r_{ki} T_i \leq \tau_3 \quad (k \% 3 = 3) \\
 & && 0 < \sum_i r_{ki} \sigma_i \leq 1 \quad (\forall k) \\
 & && 0 \leq \sum_i r_{ki} (1 - \sigma_i) \leq 3 \quad (\forall k) \\
 & && \sum_{k=1}^{3D} r_{ki} \leq 1
 \end{aligned}$$

調理時間の最小化

食材コストの最小化

摂取栄養量の制約

摂取カロリーの制約

調理時間の制約(朝)

調理時間の制約(昼)

調理時間の制約(夜)

主菜の制約

副菜の制約

料理種類の制約

変数

対象の日数: D
 レシピの数: R
 食材の数: Q
 栄養素の数: N
 データベース上の食材数: S
 データベース上の食材番号: $d: 1, 2, 3, \dots, S$
 日の番号: $k: 1, 2, 3, \dots, 3D$
 栄養素の番号: $l: 1, 2, 3, \dots, N$
 材料の番号: $m: 1, 2, 3, \dots, Q$
 レシピの番号: $i: 1, 2, 3, \dots, R$
 i 番目のレシピの名前: y_i
 i 番目のレシピの献立フラグ: r_{ki}
 i 番目のレシピの主菜フラグ: σ_i
 i 番目のレシピの調理時間: T_i
 i 番目のレシピの摂取カロリー: C_i
 i 番目のレシピの食材コスト: $G_i = \sum_{m=0}^Q \sum_{d=1}^{q_{im}} \frac{e_{im}}{W_d} M_d$
 i 番目のレシピの m 番目の材料の名前: q_{im}
 i 番目のレシピの m 番目の材料量: e_{im}
 i 番目のレシピの l 番目の栄養素の名前: n_{il}
 i 番目のレシピの l 番目の栄養素の量: f_{il}
 d 番目の食材名: Z_d
 d 番目の食材の販売単位: W_d
 d 番目の食材の値段: M_d

ログイン情報

使用者の名前: ϵ
 身長: α
 体重: β
 基礎代謝量(下限): B^L
 基礎代謝量(上限): B^H

図20 献立作成における多目的最適化の定式化

食材の購入と嗜好を考慮したデータの選別

8/20

1. はじめに
2. 自動献立作成の概要
3. 提案手法
4. おさらい
5. 数値実験
6. おわりに
7. おさらい

先行研究では、過去に購入した余剰食材を考慮しておらず、食材購入に関する考慮がされていない。そこで、本研究では余剰食材について考慮できるようにデータを選別する。

手法

2つの文字列の類似度を計算する「ゲシュタルトパターンマッチング」を用いる。

$$D_{ro}(S_1, S_2) = \frac{2K_m}{|S_1| + |S_2|} \quad (1)$$

K_m : マッチした文字数, $|S_1|, |S_2|$: それぞれの文字列の長さ

手順

- 1 ユーザに入りたい食材、または冷蔵庫にある食材を入力してもらう
- 2 レシピのデータベースのすべてのレシピと比較し、今持っている食材が含まれたレシピを新しいデータベースに蓄積
- 3 新しいデータベースから献立を選択していく

- 1. はじめに
- 2. 自動献立作成の概要
- 4. 提案手法
- 5. おさらい
- 5. 数値実験
- 5. おわりに
- 5. おさらい

手順

- 1 多目的最適化問題において大量のデータを扱う場合、処理速度が低下してしまうため、処理速度の向上に務める必要がある。

1. はじめに
2. 自動献立作成の概要
4. 提案手法
5. おさらい
5. 数値実験
5. おわりに
5. おさらい

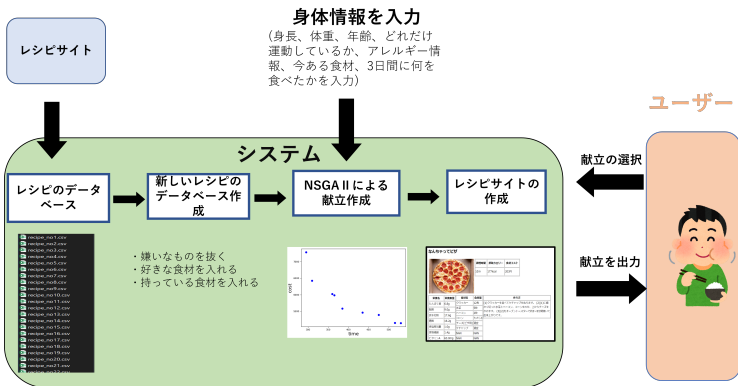


図3 システムの流れ

- 1. はじめに
- 2. 自動献立作成の概要
- 4. 提案手法
- 5. おさらい
- 5. 数値実験
- 6. おわりに
- 5. おさらい

・あ

- 1. はじめに
- 2. 自動献立作成の概要
- 4. 提案手法
- 5. おさらい
- 5. 数値実験
- 5. おわりに
- 5. おさらい

出力結果

献立作成システムをより実用化する。

- ・ ユーザの好みにあった献立を出力できるようにする
- ・ 今持っている食材が含まれるレシピを出力できるようにする
- ・ 並列分散などを用いてシステムを高速化できるようにする

- 1. はじめに
- 2. 自動献立作成の概要
- 4. 提案手法
- 5. おさらい
- 5. 数値実験
- 5. おわりに
- 5. おさらい

並列分散の結果

献立作成システムをより実用化する。

- ・ ユーザの好みにあった献立を出力できるようにする
- ・ 今持っている食材が含まれるレシピを出力できるようにする
- ・ 並列分散などを用いてシステムを高速化できるようにする

1. はじめに
2. 自動献立作成の概要
3. 提案手法
4. おさらい
5. 数値実験
6. おわりに
5. おさらい

まとめ

- 献立の出力

今後の課題

- S

- 1. はじめに
- 2. 自動献立作成の概要
- 4. 提案手法
- 5. おさらい
- 5. 数値実験
- 6. おわりに
- 5. おさらい

卒論の課題

献立作成システムをより実用化する。

- ・ ユーザの好みにあった献立を出力できるようにする
- ・ 今持っている食材が含まれるレシピを出力できるようにする
- ・ 並列分散などを用いてシステムを高速化できるようにする

食材の追加

- ・ユーザの好みにあった献立を出力できるようにする
- ・今持っている食材が含まれるレシピを出力できるようにする

- ・入れる食材を入力するテキストボックスを追加
- ・食材を追加するとその食材が含まれるレシピが出力できるようにした。
- ・複数を入力すると複数の食材が含まれたレシピが出力される

・アレルギーの情報

☒ 特になし

- ☐ 卵 ☐ 乳 ☐ らっかせい ☐ えび ☐ 小麦 ☐ かき ☐ そば
☐ アーモンド ☐ あわび ☐ いか ☐ いくら ☐ オレンジ ☐ カシューナッツ
☐ キウイフルーツ ☐ 牛肉 ☐ くるみ ☐ ごま ☐ 鮭(さけ) ☐ さば ☐ 大豆 ☐ 鶏肉
☐ バナナ ☐ 豚肉 ☐ まつたけ ☐ 桃 ☐ やまいも ☐ リンゴ ☐ ゼラチン

・なにか抜いてほしい食材があれば入力してください

・患っている、または予防したい生活習慣病

☒ とくになし ☐ 糖尿病 ☐ 腎臓病 ☐ 脂質異常症 ☐ 高血圧

・なにか入れたい食材があれば入力してください(複数入力可)

並列処理による処理速度の向上

マルチプロセスとマルチスレッドによる並列処理を行い、処理速度の向上を検討

- ・マルチスレッドは、1つの処理を細かい処理に分割して逐次に処理をする方式
- ・マルチプロセスは、複数の処理を同時に処理する方式

シングルスレッドの場合

CPU1	事前処理	処理A	処理B	処理C	事後処理
------	------	-----	-----	-----	------

マルチスレッドの場合

CPU1	事前処理	処理待ち				事後処理
CPU2						
CPU3						
CPU4						

マルチプロセスの場合

CPU1	事前処理	処理待ち	事後処理
CPU2		処理A	
CPU3		処理B	
CPU4		処理C	

1. はじめに
2. 自動献立作成の概要
3. 提案手法
4. おさらい
5. 数値実験
6. おわりに
5. おさらい

1. はじめに
2. 自動献立作成の概要
3. 提案手法
4. おさらい
5. 数値実験
6. おわりに
5. おさらい

結果

マルチスレッド

→実装はできたがスレッド数を増やしてもほとんど処理時間の変化がなかった。

マルチプロセス

→調整中、中間までには実装する。

- 1. はじめに
- 2. 自動献立作成の概要
- 4. 提案手法
- 5. おさらい
- 5. 数値実験
- 6. おわりに
- 5. おさらい

まとめ

- 入れたい食材を追加できるようにした
- 並列化による処理速度の向上を試した。

今後の課題

- ほかの高速化の手段があるか見つける

Temporary page!

\LaTeX was unable to guess the total number of pages correctly. As there was some unprocessed data that should have been added to the final page this extra page has been added to receive it.

If you rerun the document (without altering it) this surplus page will go away, because \LaTeX now knows how many pages to expect for this document.