血小板による止血機構に基づいた最適化手法

野城 滉司　長谷部 浩二

本研究では，血小板による止血機構を模倣した最適化手法を提案する．血小板による止血 (一次止血) では，血小板が傷害部位に粘着することで活性化し，他の血小板と凝集して血栓が形成される．活性化した血小板が放出する活性化因子や血管内皮細胞で産生される活性化抑制因子によって，止血は制御される．以上のような血小板の凝集，活性化の制御をモデル化し，最適化手法として用いる．またメタヒューリスティクスのベンチマークとして知られる 36 個の関数を用いて他の最適化手法と比較することにより，提案手法の評価を行う．その結果，平均して 23.3 個の関数で最適値が得られ，既存の最適化手法と比べ遜色のない性能を有することが示された．

1　はじめに

工業製品の設計やスケジューリングなど，実世界の問題の多くは非線形の最適化問題として定式化される．これらの問題は，その複雑さから解析的に解くのが困難であるため，近似的な解を求めるメタヒューリスティクスが提案されてきた．メタヒューリスティクスには，遺伝的アルゴリズム (GA) [6] や進化戦略 (ES) [1] などの生物の進化から着想を得たもの，また粒子群最適化 (PSO) [9] や蟻コロニー最適化 (ACO) [4] などの群知能と呼ばれる群れの行動を模したものなどがある．

メタヒューリスティクスには探索 (exploration) と活用 (exploitation) が要請される．Chen ら [2] は，探索を「過去にサンプリングした点の情報に依らずに，新たなサンプリングをすること」，活用を「過去にサンプリングした点の情報に依って，新たなサンプリングをすること」と定義している．これらはトレードオフの関係にあり，適切なバランスを取ることが重要である．生物の群れの行動において，個体の拡散と集約が観察でき，これを模倣したアルゴリズムは探索と活用のバランスを持った最適化を実現している．

止血は動物の持つ非常に洗練された機能であり，血管壁が傷害されると迅速に止血が開始され出血を止める．これは，平時には血小板が血管内で均等に分布しており，出血時に傷害部位に素早く凝集することによる．すなわち血小板が拡散と集約を行うことで迅速な止血を実現している．本論文では，止血機構を模倣したメタヒューリスティクスを提案する．止血機構は，血小板による一次止血と凝固系による二次止血に分類されるが，ここでは特に一次止血のメカニズムを応用したアルゴリズムを提案する．本アルゴリズムは血小板の活性化，凝集，非活性化により最適化を行う．生成された解は血小板の位置を表し，血流ベクトルに従って移動する．全ての血小板の中で最適解となる血小板が活性化し，その周辺にある血小板を活性化させる．活性化した血小板は最適解となる血小板に引き付けられ，凝集する．活性化した血小板が一定数を超えると，全ての血小板は非活性となる．以上を繰り返すことで，局所解に収束するのを防ぎながら最適解を探索する．

また本論文では，提案手法の性能を評価するため，

Optimization algorithm based on platelet plug formation mechanism

Koji Noshiro, 筑波大学情報学群情報科学類, College of Information Science, University of Tsukuba.

Koji Hasebe, 筑波大学システム情報系, Faculty of Engineering, Information and Systems, University of Tsukuba.

メタヒューリスティクスのベンチマークとして知られる 36 個の関数[5] を用いて実験を行った．関数の評価の最大回数を 100,000 回，300,000 回，500,000 回とする 3 通りの条件下で 30 回ずつ最適値を求めると，真の最適値を得られた関数の数は平均 23.3 個であった．また求めた最適値の平均値を粒子群最適化 (PSO)，遺伝的アルゴリズム (GA)，蟻コロニー最適化 (ACO) と比較した．一般に次元の高い関数については優位性を認められなかったものの，特に 4 次元以下の次元の低い関数ではこれらのアルゴリズムと比べ遜色のない性能を有することが示された．また単峰性関数と多峰性関数のどちらであっても同等の結果が得られ，良い探索の性能を有すると言える．

本論文の構成は以下の通りである．第 2 章で止血機構の概要を示す．第 3 章では提案手法について説明し，第 4 章でその評価を述べる．第 5 章では関連研究を紹介し，第 6 章で結論と今後の課題について述べる．

2 止血機構の概要

止血とは，出血が起きた際，血小板や凝固機構が作用することで止血血栓を形成し，出血を防ぐことである (詳細は文献[18] などを参照のこと)．止血は，活性化した血小板が粘着，凝集することによって血小板血栓を形成する一次止血と，凝固系の反応によってより強固なフィブリン血栓を形成する二次止血に分類される．止血機構は増幅系であり，少量の凝固因子から大量の凝固因子を産生する．また同時に制御因子も産生され，過剰な血栓形成を防ぐ．

2.1 一次止血

血管内皮の傷害によって露出された血管内皮下組織のコラーゲンと血小板が触れることにより，止血機構は開始する．コラーゲンと粘着した血小板は活性化し，血小板内部にある顆粒を放出する．この顆粒は自身や周囲の血小板の活性化，また凝固系の活性化を促進させる．活性化した血小板は互いに結合し，凝集する．これにより血小板血栓を形成する．正常血管内皮細胞で放出される PGI_2 や NO によって，血小板の活性化は抑制される．また血小板が凝集する際にも NO は産生され，血小板自身が活性化の抑制に寄与する．

2.2 二次止血

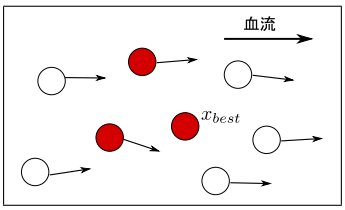
二次止血は外因系と内因系に分けられる．これらは開始機構が異なる．外因系は，血管外組織などに発現する組織因子が，出血により血液と触れ，血液中の成分と結合することで開始される．内因系は，血液が内皮下組織や異物に触れることで開始される．このような初期反応が開始すると，凝固カスケードと呼ばれる連鎖的な反応を起こし，凝固因子を活性化させ，また活性化を増幅させる．凝固カスケードは最終的にトロンビンという酵素を産生し，血漿中の可溶性のフィブリノーゲンを不溶性のフィブリンに変化させ，強固なフィブリン血栓を形成する．これらの反応には活性化血小板の膜表面に露出するリン脂質が必要であり，血小板の活性化は二次止血に大きく関わる．また凝固因子の活性化により，凝固制御因子や線溶因子を活性化させ，不要な血栓の形成の防止およびその溶解を行う．

3 止血機構に基づいた最適化手法

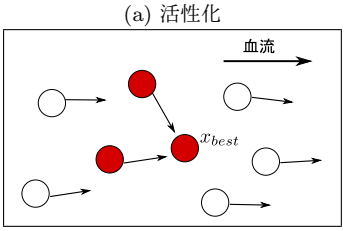
本章では，止血機構 (特に一次止血) を模倣したアルゴリズムについて述べる．

3.1 アルゴリズムの概要

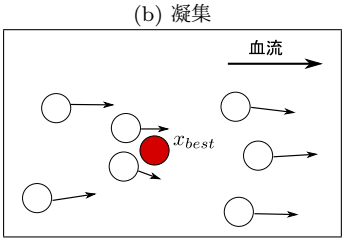
本アルゴリズムの概略図を図 1 に示す．第 2 章で述べたように，一次止血では，血小板が傷害部位に粘着し，活性化を起こす．活性化した血小板は周囲の血小板を活性化させ (図 1a)，それらは凝集し血小板血栓を形成する (図 1b)．本アルゴリズムでは，解 $\boldsymbol{x}$ が血小板の位置を表し，その時点における最適解 $\boldsymbol{x_{best}}$ を傷害部位とみなす．また現実の生体において，血小板血栓の形成に血小板が使われても，血流により血小板が常に供給され，血小板の数は一定となる．これをモデル化するため，本アルゴリズムでは便宜的に，血小板血栓の形成が終わると血栓を構成する血小板は凝集をほどき，血小板の数が元に戻るものとする．すなわち活性化血小板が十分に増えたとき，血小板は非活性化される (図 1c)．この仮定は過剰な血小板の活性化を防ぐという性質のモデル化にもなっている．



(a) 活性化



(b) 凝集



(c) 非活性化

図 1: 本アルゴリズムの概略図

以上のような血小板の活性化と凝集，非活性化によって最適化を行う．

本アルゴリズムの疑似コードを Algorithm 1 に示す．本アルゴリズムは以下のような流れで処理が進む．

Step1. ランダムに血小板の位置を生成する．また全ての血小板は非活性とする． (疑似コードの 2，3 行目)

Step2. 活性化血小板の数が閾値を超えていた場合，全ての血小板を非活性にする． (8-10 行目)

Step3. 最適解となる血小板を活性化する． (8-10 行目)

Step4. 非活性化血小板は血流ベクトルに従って移動し，活性化血小板は最適解となる血小板に引き寄せられる． (11-15 行目)

Step5. 活性化血小板の周囲の非活性化血小板を活性化する． (17-28 行目)

Step6. イテレーションが終了条件に満たない場合，Step2. に戻る． (30 行目)

Step7. 最適解を出力する． (31 行目)

Algorithm 1 止血機構に基づいたアルゴリズム

```
    //N ：解の数，Max_iteration ：最大のイテレーション
 1: α, β, γ, δ, ϵ の設定
 2: x の初期化
 3: a の初期化
 4: F の初期化 (式 1)
 5: while iteration ≤ Max_iteration do
 6:     f(x) の計算
 7:     x_best の更新
 8:     if a_best = 0 ∨ Σ_j a_j ≥ ϵ then
 9:         a の更新 (式 4)
10:     end if
11:     for i = 1 to N do
12:         v_f の計算 (式 2)
13:         x_i の更新 (式 3)
14:         制約条件を満たすように x_i を修正
15:     end for
16:     A ← ∅
17:     for k = 1 to N do
18:         if a_k = 1 then
19:             for l = 1 to N do
20:                 if a_l = 0 then
21:                     d_kl ← |x_k − x_l|_2
22:                 end if
23:             end for
24:             d_kl を昇順に並べる
25:             A の要素を追加 (式 6)
26:         end if
27:     end for
28:     a の更新 (式 5)
29:     iteration ← iteration + 1
30: end while
31: x_best を出力
```

以下，3.2 節では Step4 の血小板の移動で用いる血流ベクトルと血流速度を説明し，3.3 節で血小板の移動について述べる．3.4 節では Step2，Step3，Step5

における血小板の活性化状態の更新について説明する．

3.2 血流ベクトルと血流速度

アルゴリズムの開始時に N 個の解 $\boldsymbol{x_1}, \cdots, \boldsymbol{x_N}$ がランダムに生成される．これらはそれぞれ血小板 $1, \cdots,$ 血小板 N の位置を表す．血小板の総数 N はアルゴリズムを通して一定である．

血流の方向を表す血流ベクトルを $\boldsymbol{F}$ とする．$\boldsymbol{F}$ はアルゴリズムの初期に乱数によって決定される．$\boldsymbol{F}$ の第 d 成分は以下のように与えられる．

$$F_d = rand() \in [-1, 1] \tag{1}$$

ここで，$rand()$ は乱数を生成する関数である．

また血小板が血流ベクトルに従って移動する量を血流速度 v_f とする．流体の性質から，血管壁から離れた部分よりも血管壁に近い部分の方が血流速度が小さくなる．すなわち傷害部位に近い部分は，血流が遅く流れる．よって最適解となる血小板の位置 $\boldsymbol{x_{best}}$ に近い血小板は遅く移動するとする．ある血小板 i の血流速度 v_f は以下のように与えられる．

$$v_f = |\boldsymbol{x_i} - \boldsymbol{x_{best}}|_2 + rand() \tag{2}$$

ここで，$rand() \in [0, 1]$ である．

3.3 血小板の位置の更新

血小板の移動は血小板の活性化状態に応じて異なる．血小板が非活性であるとき，血流ベクトルとランダムなベクトルを足した方向に移動する．血小板が活性化しているとき，血流ベクトルを無視して，最適解となる血小板の位置に近づく．血小板 i の位置 $\boldsymbol{x_i}$ の更新式は以下のように与えられる．

$$\begin{aligned} \boldsymbol{x_i}(t+1) =& \boldsymbol{x_i}(t) + (1 - a_i) \cdot \alpha v_f (\boldsymbol{F} + \beta \boldsymbol{R}) \\ &+ a_i \cdot \gamma (\boldsymbol{x_{best}} - \boldsymbol{x_i}(t)) \end{aligned} \tag{3}$$

ここで，$a_i \in \{0, 1\}$ は血小板 i の活性化状態を表す．非活性のときは $a_i = 0$，活性化しているときは $a_i = 1$ である．$\boldsymbol{R}$ はランダムなベクトルであり，第 d 成分は以下のように与えられる．

$$R_d = rand() \in [-1, 1]$$

また α, β, γ はパラメータであり，$\alpha, \beta, \gamma \in [0, 1]$ である．α は血流速度の大きさを調整する．β は血流ベクトルに対する乱数の影響の大きさを決め，この値が大きいほど非活性化血小板はランダムに動く．また γ は活性化血小板の $\boldsymbol{x_{best}}$ への近づき方を表し，この値が大きいほど $\boldsymbol{x_{best}}$ の近くに移動する．

3.4 活性化状態の更新

血小板の活性化状態 a は 0 か 1 の 2 値を取り，$a = 0$ のときは非活性，$a = 1$ のときは活性化していることを表す．活性化状態の更新は，位置の更新の前後で 2 度行う．

1 度目の活性化状態の更新は，最適解を取る血小板を活性化させる．最適解となる血小板が非活性であった場合，その血小板は活性化し，他の血小板を全て非活性にする．また活性化血小板の総数が ϵ 個以上の場合も，他の血小板を全て非活性にする．最適解を取る血小板を血小板 $best$，更新前の血小板 i の活性化状態を $a'_i(t)$ と表すと，血小板 i の 1 度目の活性化状態の更新式は次のように与えられる．

$$a_i(t) = \begin{cases} 1 & \text{if } i = best \wedge C(t) \\ 0 & \text{if } i \neq best \wedge C(t) \\ a'_i(t) & \text{otherwise} \end{cases} \tag{4}$$

ここで，

$$C(t) \equiv a'_{best}(t) = 0 \vee \sum_j a'_j(t) \geq \epsilon$$

とする．

2 度目の更新後の活性化状態は，活性化血小板との距離関係によって決まる．更新前に活性化している血小板は更新後も活性化する．更新前に活性化している血小板 k は，距離が近い δ 個の非活性化血小板を活性化させる．この操作を全ての活性化血小板で行う．血小板 i の活性化状態 a_i の更新式は次のように与えられる．

$$a_i(t+1) = \begin{cases} 1 & \text{if } i \in \boldsymbol{A} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

ここで，$\boldsymbol{A}$ は次のように定義する．

$a_k(t) = 1, a_l(t) = 0$ なる k, l が存在し，$d_{k,l} = |x_k - x_l|_2$ とする．$d_{k,l_1} \leq \cdots \leq d_{k,l_\delta} \leq \cdots \leq d_{k,l_M}$ のように $d_{k,l}$ を昇順に並べる．ここで，M は非活性化血小板の個数とする．このとき，

$$l_1, \cdots, l_\delta, k \in \boldsymbol{A} \tag{6}$$

これを全ての k について適用する．

δ, ϵ はパラメータであり，$\delta \in [0, \epsilon], \epsilon \in [0, N]$ である．δ は 1 つの活性化血小板が活性化させる非活性化血小板の個数を表し，ϵ は活性化血小板の個数の上限を表す．

4 実験と評価

本研究では，第 3 章で説明した提案手法の性能を評価するため，数値実験を行った．本章では，その実験結果について述べる．本研究では，Ezugwu ら [5] によって提案されたメタヒューリスティクスの包括的な評価方法に従うものとする．

4.1 実験方法

Ezugwu ら [5] によって提案された評価方法について説明する．従来の評価方法ではイテレーションの数を基準にすることが多いが，イテレーションに対する意味や計算回数はアルゴリズムごとに異なる．そのためイテレーションの数の代わりに，関数の評価の回数をもとにアルゴリズムを評価する．生成する解の数を 50 とし，関数の評価の最大回数を 100,000 回，300,000 回，500,000 回とする．目的関数は異なる性質を持つ F1-F36 と名付けられた 36 個の関数を用いる．関数の定義や性質については文献 [5] を参照されたい．1 つの目的関数に対して 30 回ずつアルゴリズムを実行し，最適値を求める．なお提案された方法では，実行時間の計測やアルゴリズム間の統計分析を行っているが，本研究では行わないこととする．また以上に加え，本アルゴリズムの収束性についても評価を行う．パラメータは $\alpha = 1.0, \beta = 0.8, \gamma = 0.9, \delta = 3, \epsilon = 45$ と設定する．

表 1: 真の最適値を得られた関数の数

関数評価の回数	100,000	300,000	500,000	平均
DE [13]*	29	30	30	29.7
PSO [9]*	29	29	29	29
GA [6]*	29	29	29	29
CS [17]*	22	30	30	27.3
SOS [3]*	26	28	26	26.7
ACO [4]*	21	29	29	26.3
本アルゴリズム	**21**	**24**	**25**	**23.3**
FPA [16]*	20	23	24	22.3
IWO [10]*	20	20	19	19.7
BA [15]*	22	19	17	19.3
BeeA [12]*	15	20	20	18.3
ABC [8]*	15	16	21	17.3
FA [14]*	17	18	16	17

* 文献 [5] より引用．

4.2 結果と考察

アルゴリズムを 30 回実行して最適値を求め，真の最適値が得られた関数の数を表 1 に示す．結果の比較のため，文献 [5] において比較された 12 個のアルゴリズムの結果を引用している．本アルゴリズムは，関数の評価の最大回数が増えると真の最適値が得られる数も増えている．また PSO や GA には劣るものの，ABC や FA など 5 つのアルゴリズムに対しては，関数の評価の最大回数がいずれの場合においても，本アルゴリズムの方が優れている．

また得られた最適値の平均値と標準偏差を表 2，3，4 に示す．Mean が平均値，SD が標準偏差を表す．また関数の性質について，U は単峰性，M は多峰性であることを表し，その右の値は次元を表す．これらの表では文献 [5] に示された PSO，GA，ACO の結果を引用している．得られた最適値の平均値について，4 つのアルゴリズムの中で最小値を取っているものは太字で表し，その個数も示す．本アルゴリズムは，関数の評価の最大回数が 100,000 回のときは 16 個の関数で，300,000 回と 500,000 回は 19 個の関数で最小値を取っている．その中の多くは他のアルゴリズムも

表 2: 関数の評価の最大回数が 100,000 回のとき

		本アルゴリズム		PSO*		GA*		ACO*	
関数	性質	Mean	SD	Mean	SD	Mean	SD	Mean	SD
F1	M2	**0**	0	**0**	0	2.45E-05	6.56E-05	**0**	0
F2	M2	**-1**	0	**-1**	0	**-1**	0	**-1**	0
F3	U2	**0**	0	**0**	0	3E-08	8.53E-08	**0**	0
F4	U2	**0**	0	**0**	0	**0**	0	**0**	0
F5	U2	**0**	0	0.118356	0.218206	0.237458	0.258202	0.373265	0.196866
F6	M2	**-1.8013**	0	**-1.8013**	0	**-1.8013**	0	**-1.8013**	0
F7	U2	**0**	0	**0**	0	0.01019	0.018787	**0**	0
F8	U2	**-1.0316**	0	**-1.0316**	0	**-1.0316**	0	**-1.0316**	0
F9	U2	**0**	0	**0**	0	**0**	0	**0**	0
F10	M2	8.23E-09	4.43E-08	**0**	0	7.79E-05	0.00034	**0**	0
F11	M2	**-186.731**	0	**-186.731**	0	**-186.731**	0	-186.56	0.163398
F12	M4	0.295696	1.406533	**3.55E-07**	5.84E-07	0.389614	0.366744	0.003802	0.003737
F13	M5	-4.240032	0.396426	-4.53059	0.141476	**-4.6877**	0	-4.61488	0.232736
F14	U30	5.15E-08	1.57E-07	**0**	0	**0**	0	**0**	0
F15	M30	-7.154645	0.982979	-8.89867	0.454675	**-9.65878**	0.007627	-5.42275	0.609768
F16	U30	3.93E-09	4.06E-09	**0**	0	2.98E-08	9.2E-08	2.28E-05	1.19E-05
F17	U30	2.78E-05	4.99E-05	**0**	0	9.46E-06	3.17E-05	0.008633	0.004304
F18	U30	0.019778	0.017518	**0**	0	6.12E-07	2.53E-06	0.001204	0.00058
F19	U30	0.355379	0.167819	0.005325	0.001946	**0.000557**	0.000319	0.101121	0.026612
F20	U30	49.52619	54.59519	0.005464	0.017385	**1.14E-06**	6.22E-06	37.53509	30.31922
F21	U30	4.451599	1.602339	**0**	0	2.58E-05	8.89E-05	0.11226	0.079245
F22	U30	38.30046	25.30333	26.04939	22.67168	**7.034444**	11.7931	3133.414	2722.962
F23	U30	0.741405	0.120788	0.66667	0	**0.571138**	0.299638	13.25126	6.730554
F24	M30	148.2597	45.41950	46.16602	11.62541	**2.23E-05**	5.14E-05	241.0779	16.92551
F25	U30	0.267006	0.212181	0.02721	0.027512	**0.019158**	0.0139	0.194886	0.257887
F26	M30	6.28151	0.965155	0.979077	0.723107	**0.000382**	0.000751	0.027602	0.014651
F27	M2	-0.997875	0.011444	**-1**	0	-0.9915	0.022041	**-1**	0
F28	M3	**-3.8628**	4.93E-07	**-3.8628**	0	**-3.8628**	0	**-3.8628**	0
F29	M10	-3.269323	0.052723	**-3.3224**	0	-3.28664	0.055558	**-3.3224**	0
F30	M4	**-8.045458**	2.615378	-5.97419	3.371298	-7.40338	3.491636	-5.15602	3.529503
F31	M4	**-8.67691**	2.689388	-7.91495	3.398333	-8.17576	3.476117	-8.41198	3.00258
F32	M4	**-9.99864**	1.612921	-7.66899	3.841268	-8.43545	3.30999	-9.33238	2.754959
F33	M2	**0.39789**	0	**0.39789**	0	**0.39789**	0	**0.39789**	0
F34	M2	**3**	0	**3**	0	**3**	0	**3**	0
F35	M5	0.123207	0.049554	**0.099873**	0	0.106539	0.02537	**0.099873**	0
F36	M10	0.11654	0.037268	**0.099873**	0	0.183204	0.069892	**0.099873**	0
Min		16 個		23 個		19 個		17 個	

* 文献 [5] より引用.

表 3: 関数の評価の最大回数が 300,000 回のとき

		本アルゴリズム		PSO*		GA*		ACO*	
関数	性質	Mean	SD	Mean	SD	Mean	SD	Mean	SD
F1	M2	**0**	0	**0**	0	3.87E-06	7.45E-06	**0**	0
F2	M2	**-1**	0	**-1**	0	**-1**	0	**-1**	0
F3	U2	**0**	0	**0**	0	7.24E-08	3.07E-07	**0**	0
F4	U2	**0**	0	**0**	0	**0**	0	**0**	0
F5	U2	**0**	0	0.4227	0.192269	0.152587	0.237066	0.343503	0.214024
F6	M2	**-1.8013**	0	**-1.8013**	0	**-1.8013**	0	**-1.8013**	0
F7	U2	**0**	0	**0**	0	0.018924	0.02201	**0**	0
F8	U2	**-1.0316**	0	**-1.0316**	0	**-1.0316**	0	**-1.0316**	0
F9	U2	**0**	0	**0**	0	**0**	0	**0**	0
F10	M2	**0**	0	**0**	0	4.94E-05	0.000127	**0**	0
F11	M2	**-186.731**	0	**-186.731**	0	**-186.731**	0	-186.688	0.038565
F12	M4	0.000356	0.000863	**0**	0	0.377487	0.245328	1.65E-06	1.57E-06
F13	M5	-4.396389	0.234244	-4.5377	0	**-4.6877**	0	-4.64941	0.171731
F14	U30	**0**	0	**0**	0	0.010732	0.005171	**0**	0
F15	M30	-7.446424	0.937955	-8.97165	0.43584	**-9.6602**	0	-5.72682	0.650794
F16	U30	**0**	0	**0**	0	1.94E-09	5.52E-09	**0**	0
F17	U30	8.47E-08	1.53E-07	**0**	0	3.9E-07	1.49E-06	**0**	0
F18	U30	8.14E-05	0.000126	**0**	0	1.5E-06	5.48E-06	**0**	0
F19	U30	0.461125	0.281962	0.002083	0.000977	**0.000199**	0.000118	0.025673	0.008558
F20	U30	16.61293	37.49047	0.007818	0.016099	2.74E-06	8.07E-06	**1.75E-08**	5.78E-08
F21	U30	0.060863	0.044624	**0**	0	1.7E-05	5.07E-05	**0**	0
F22	U30	34.08591	24.05140	31.15628	24.47534	**4.559312**	3.56795	24.81089	0.201134
F23	U30	0.682474	0.043712	0.66667	0	**0.440574**	0.340876	0.66667	0
F24	M30	121.5057	44.00579	25.37472	0.235326	**2.33E-06**	4.49E-06	216.2763	14.33827
F25	U30	**0.012905**	0.007227	0.013267	0.017928	0.025758	0.017195	0.024944	0.065464
F26	M30	5.838586	0.999047	1.087033	0.813943	0.000199	0.000419	**2.45E-09**	1.26E-09
F27	M2	**-1**	0	**-1**	0	**-1**	0	**-1**	0
F28	M3	**-3.8628**	0	**-3.8628**	0	**-3.8628**	0	**-3.8628**	0
F29	M10	-3.253156	0.053453	**-3.3224**	0	**-3.3224**	0	-3.28267	0.057152
F30	M4	-8.631395	2.324628	-5.8888	3.423882	-7.23869	3.656347	**-10.1532**	0
F31	M4	**-9.696981**	1.799865	-7.50239	3.656702	-8.58732	3.104273	-9.09395	2.698397
F32	M4	-9.641377	2.001369	-7.78822	3.70402	-9.36999	2.687304	**-9.74316**	2.147481
F33	M2	**0.39789**	0	**0.39789**	0	**0.39789**	0	**0.39789**	0
F34	M2	**3**	0	**3**	0	**3**	0	**3**	0
F35	M5	0.099873	0	0.099873	0	0.103206	0.018257	**0.087597**	0.021277
F36	M10	0.11654	0.037268	0.173204	0.058328	0.163205	0.061493	**0.099873**	0
Min		19 個		21 個		17 個		24 個	

* 文献 [5] より引用.

表 4: 関数の評価の最大回数が 500,000 回のとき

		本アルゴリズム		PSO*		GA*		ACO*	
関数	性質	Mean	SD	Mean	SD	Mean	SD	Mean	SD
F1	M2	**0**	0	**0**	0	2.05E-06	3.28E-06	**0**	0
F2	M2	**-1**	0	**-1**	0	**-1**	0	**-1**	0
F3	U2	**0**	0	**0**	0	5.9E-08	1.35E-07	**0**	0
F4	U2	**0**	0	**0**	0	**0**	0	**0**	0
F5	U2	**0**	0	**0**	0	0.2199	0.255764	**0**	0
F6	M2	**-1.8013**	0	**-1.8013**	0	**-1.8013**	0	**-1.8013**	0
F7	U2	**0**	0	**0**	0	0.013101	0.020355	**0**	0
F8	U2	**-1.0316**	0	**-1.0316**	0	**-1.0316**	0	**-1.0316**	0
F9	U2	**0**	0	**0**	0	**0**	0	**0**	0
F10	M2	**0**	0	**0**	0	4.4E-06	1.46E-05	**0**	0
F11	M2	**-186.731**	0	**-186.731**	0	**-186.731**	0	-186.706	0.023844
F12	M4	0.000345	0.001822	**0**	0	0.252108	0.248563	1.74E-09	4.39E-09
F13	M5	-4.399716	0.286198	-4.51818	0.17518	-4.6877	0	**-4.7187**	0.095547
F14	U30	**0**	0	**0**	0	0.004059	0.001878	**0**	0
F15	M30	-7.663725	0.93645	-8.8309	0.549215	**-9.6602**	0	-6.14073	0.618152
F16	U30	**0**	0	**0**	0	1.66E-09	3.09E-09	**0**	0
F17	U30	2.72E-08	3.29E-08	**0**	0	1.09E-06	4.09E-06	**0**	0
F18	U30	1.85E-06	2.29E-06	**0**	0	1.14E-08	2.77E-08	**0**	0
F19	U30	0.322994	0.198547	0.003114	0.008358	**0.000128**	7.84E-05	0.015298	0.004101
F20	U30	24.62230	45.15849	0.003114	0.008358	7.32E-07	3.87E-06	**0**	0
F21	U30	0.00319	0.003636	**0**	0	2.55E-06	8.92E-06	**0**	0
F22	U30	38.12486	35.29815	28.80641	22.78898	**4.987585**	3.966018	21.53581	0.223952
F23	U30	0.666935	0.000649	0.66667	0	**0.555667**	0.252455	0.66667	0
F24	M30	122.7093	42.99902	51.63829	16.08217	**1.54E-06**	1.75E-06	210.3432	8.29743
F25	U30	0.010114	0.0083	0.010661	0.01356	0.024197	0.023658	**0**	0
F26	M30	5.69971	2.608923	0.8089	0.873708	0.000154	0.000316	**0**	0
F27	M2	**-1**	0	**-1**	0	-0.98513	0.027424	**-1**	0
F28	M3	**-3.8628**	0	**-3.8628**	0	**-3.8628**	0	**-3.8628**	0
F29	M10	-3.253846	0.05276	**-3.8628**	0	-3.27869	0.058424	-3.3224	0
F30	M4	**-7.951661**	2.517598	-5.38903	3.098941	-6.48991	3.751506	-6.66531	3.792462
F31	M4	**-9.521178**	1.971699	-7.56765	3.577358	-8.39645	3.38837	-9.51602	2.091533
F32	M4	-9.461778	2.147016	-7.88964	3.809105	-9.17502	2.7908	**-10.5364**	0
F33	M2	**0.39789**	0	**0.39789**	0	**0.39789**	0	**0.39789**	0
F34	M2	**3**	0	**3**	0	**3**	0	**3**	0
F35	M5	0.099873	0	0.099873	0	0.099873	0	**0.09033**	0.014239
F36	M10	0.103207	0.017951	**0.099873**	0	0.199871	0.064326	**0.099873**	0
Min		19 個		23 個		14 個		26 個	

* 文献[5] より引用.

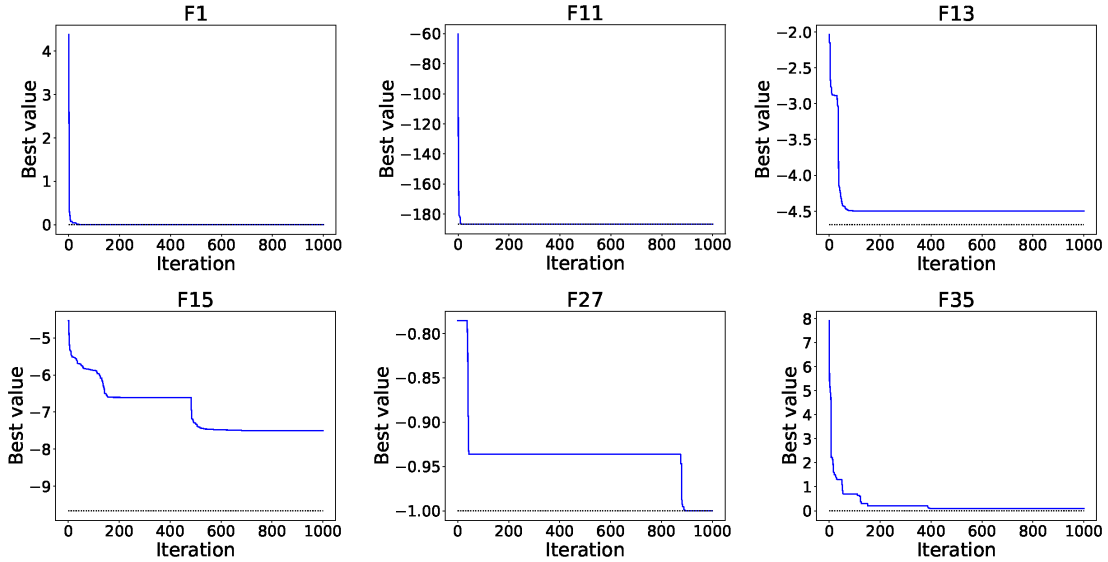


図 2: 本アルゴリズムの収束性

真の最適値を取っているが，F31 などいくつかの関数で，他よりも優れた結果を出している．また最小値を取っていない関数であっても，最小値に近い値となるものが多い．関数の性質に関して，一般に多峰性および次元の高い関数は最適化が困難である．本アルゴリズムは，単峰性と多峰性のどちらの関数でも最小値またはそれに近い値を取っており，探索という観点で良い性能を有すると言える．また関数の次元に着目すると，本アルゴリズムが最小値を取っているのは，F14 と F16 のみが 30 次元であり，それ以外は 4 次元以下である．したがって次元の高い関数に対しては，次元の低いものと比べると優位性が低くなると言える．

次に，本アルゴリズムの収束性について議論する．関数 F1，F11，F13，F15，F27，F35 について，イテレーションの進行と得られた最適値の関係を図 2 に示す．点線は真の最適値を表す．F1 と F11 は序盤で最適値となっており，早い段階で収束している．F13 もすぐに収束するが，その解は局所解となっている．F15，F27，F35 において，一定の値を維持した後，急激に最適値が小さくなるという階段状の変化が見られる．この様子から，一時的に局所解に留まっており，探索によってより良い解を発見していると考えられる．探索は，非活性化血小板が血流ベクトル $\boldsymbol{F}$ に従って移動する中で行われる．式 2 で表されるように，現在の最適解 $\boldsymbol{x}_{best}$ から離れた血小板ほど血流速度 v_f は大きくなる．このため，$\boldsymbol{x}_{best}$ から離れた血小板が移動した点が，現在の最適解よりも良い解となる可能性は小さい．このような理由から，F15 や F27 では局所解に留まる時間が長くなっていると考えられる．

5 関連研究

本研究の提案手法において，血小板の活性化と非活性化を制御することで探索と活用のバランスを取っているように，解の拡散と集約という操作を取り入れたメタヒューリスティクスの研究について述べる．

5.1 Ions motion algorithm

Javidy ら [7] は，イオンの運動を模倣したアルゴリズム (Ions motion algorithm，IMO) を提案している．イオンには，正の電荷を持つ陽イオンと負の電荷を持つ陰イオンがある．同符号の電荷を持つイオン同士は反発しあうが，異符号の電荷を持つイオン同士は引き付け合う．この性質を利用することで，適切な探索と活用のバランスを取って最適化を行う．解はイオンの位置を表し，それらは陽イオンと陰イオン

の 2 つに分けられる．IMO には流体フェーズと結晶フェーズがある．流体フェーズでは，各イオンは，異符号の電荷を持つイオンの中で最適解となるイオンに近づく．このフェーズにおいて，同符号の電荷を持つイオンによる影響は考えない．結晶フェーズでは，イオンの移動が確率的に決められ，イオン間の距離を保つ．また低い確率でイオンの位置が初期化される．このように，結晶フェーズは局所解への収束を防ぐ．また IMO は設定するパラメータを持たないという利点がある．

5.2 Dragonfly algorithm

Mirjalili ら [11] は，トンボの行動を模倣したアルゴリズム (Dragonfly algorithm，DA) を提案している．トンボは特徴の異なる 2 種類の群れを形成する．1 つは静的な群れである．これは狩りを目的としており，小さな集団を作り，飛ぶ経路を急激に変化させるといった行動をする．もう 1 つは動的な群れであり，多くのトンボが同じ方向に移動する．この 2 種類の群れは，メタヒューリスティクスにおける活用と探索に類似しており，DA はこれを利用する．DA におけるトンボの移動は，5 つの行動を足し合わせることで決定する．5 つの行動とは,「他の個体との衝突を避ける」,「他の個体と速度を合わせる」,「群れの中心に向かう」,「餌に向かう」,「捕食者から逃げる」である．これらを足し合わせる際，それぞれの重みを変化させることで，上述の静的な群れと動的な群れを形成することができる．最も良い解を餌の位置，最も悪い解を捕食者の位置として，トンボを移動させ，最適化を行う．単目的最適化問題だけでなく，解が 2 値である組み合わせ最適化問題や多目的最適化問題を解くための DA も提案している．

6 結論と今後の課題

本論文では，血小板による止血機構を模倣したアルゴリズムを提案した．本アルゴリズムは血小板の活性化，凝集，非活性化により，探索と活用のバランスを持った最適化を行う．またメタヒューリスティクスのベンチマークとして知られる 36 個の関数を用いて数値実験を行い，本アルゴリズムの評価をした．アルゴリズムを 30 回ずつ実行して最適値を求め，真の最適値が得られた関数の数を既存の 12 個のアルゴリズムと比べると，5 個のアルゴリズムよりも優れた結果が得られた．また求めた最適値の平均値を PSO，GA，ACO と比較した．これら 3 つのアルゴリズムと比べて，平均値が最小となった関数は平均して 18 個であった．最小とならずとも最小に近い値を取っている関数も多く，それらは単峰性と多峰性のどちらの関数にも認められ，良い探索の性能を有すると言える．また次元の低い関数については 3 つのアルゴリズムと比べ遜色のない性能を有することが示された．

今後の課題として，局所解に留まった際に，効率良くより良い解を見つける仕組みの導入や，適切なパラメータの設定がある．本アルゴリズムの評価についても，生成する解の数の違いによる性能の変化，他のアルゴリズムとの有意差，実行時間なども議論する必要がある．また本論文では，一次止血のメカニズムを応用したアルゴリズムを提案したが，これに二次止血のメカニズムを加えた 2 段階の最適化アルゴリズムを開発することも課題として挙げられる．

参 考 文 献

[1] Beyer, H.-G. and Schwefel, H.-P.: Evolution strategies–a comprehensive introduction, *Natural computing*, Vol. 1, No. 1(2002), pp. 3–52.

[2] Chen, J., Xin, B., Peng, Z., Dou, L., and Zhang, J.: Optimal contraction theorem for exploration–exploitation tradeoff in search and optimization, *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, Vol. 39, No. 3(2009), pp. 680–691.

[3] Cheng, M.-Y. and Prayogo, D.: Symbiotic organisms search: a new metaheuristic optimization algorithm, *Computers & Structures*, Vol. 139(2014), pp. 98–112.

[4] Dorigo, M., Birattari, M., and Stutzle, T.: Ant colony optimization, *IEEE computational intelligence magazine*, Vol. 1, No. 4(2006), pp. 28–39.

[5] Ezugwu, A. E., Adeleke, O. J., Akinyelu, A. A., and Viriri, S.: A conceptual comparison of several metaheuristic algorithms on continuous optimisation problems, *Neural Computing and Applications*, Vol. 32, No. 10(2020), pp. 6207–6251.

[6] Holland, J.: Adaptation in natural and artificial systems, 1975.

[7] Javidy, B., Hatamlou, A., and Mirjalili, S.: Ions motion algorithm for solving optimization problems, *Applied Soft Computing*, Vol. 32(2015), pp. 72–79.

[8] Karaboga, D.: An idea based on honey bee swarm for numerical optimization, Technical report, Technical report-tr06, Erciyes university, engineering faculty, computer ⋯, 2005.

[9] Kennedy, J. and Eberhart, R.: Particle swarm optimization, *Proceedings of ICNN'95-international conference on neural networks*, Vol. 4, IEEE, 1995, pp. 1942–1948.

[10] Mehrabian, A. R. and Lucas, C.: A novel numerical optimization algorithm inspired from weed colonization, *Ecological informatics*, Vol. 1, No. 4(2006), pp. 355–366.

[11] Mirjalili, S.: Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems, *Neural Computing and Applications*, Vol. 27, No. 4(2016), pp. 1053–1073.

[12] Pham, D. T. and Castellani, M.: The Bees Algorithm: Modelling foraging behaviour to solve continuous optimization problems, *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, Vol. 223, No. 12(2009), pp. 2919–2938.

[13] Storn, R. and Price, K.: Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, *Journal of global optimization*, Vol. 11, No. 4(1997), pp. 341–359.

[14] Yang, X.-S.: *Nature-inspired metaheuristic algorithms*, Luniver press, 2010.

[15] Yang, X.-S.: A new metaheuristic bat-inspired algorithm, *Nature inspired cooperative strategies for optimization (NICSO 2010)*, Springer, 2010, pp. 65–74.

[16] Yang, X.-S.: Flower pollination algorithm for global optimization, *International conference on unconventional computing and natural computation*, Springer, 2012, pp. 240–249.

[17] Yang, X.-S. and Deb, S.: Cuckoo Search via Lévy flights, *2009 World Congress on Nature Biologically Inspired Computing (NaBIC)*, 2009, pp. 210–214.

[18] 浦野哲盟, 後藤信哉: 血栓形成と凝固・線溶: 治療に生かせる基礎医学, メディカル・サイエンス・インターナショナル, 2013.