メタ戦略のロバスト性について

Metaheuristics as Robust and Simple Optimization Tools

柳浦 睦憲　　茨木 俊秀
Mutsunori Yagiura　Toshihide Ibaraki

京都大学工学研究科数理工学教室
〒606-01 京都市左京区吉田本町
Graduate School of Engineering, Kyoto University
Kyoto 606-01, Japan
{yagiura, ibaraki}@kuamp.kyoto-u.ac.jp

Abstract — One of the attractive features of recent metaheuristics is in its robustness and simplicity. To investigate this direction, we solved the single machine scheduling problem by various metaheuristics, such as random multi-start local search (MLS), genetic algorithm (GA), simulated annealing (SA) and tabu search (TS), using rather simple inside operators. The results indicate that: (1) MLS is usually good enough for practical purposes, considering its simplicity, (2) a variant of MLS, called GRASP (greedy randomized adaptive search procedure), is effective; however, its performance is sensitive to greedy methods used to generate initial solutions, (3) a variant of MLS, called iterated local search, is quite effective, (4) GA combined with local search is also competitive if longer computational time is allowed, and its performance is not sensitive to crossovers, (5) SA (and its variants called the threshold accepting and the great deluge algorithm) is another competitive method assuming that longer computational time is allowed, and its performance is not much dependent on inside parameter values, (6) there are cases in which TS is more effective than MLS; however, its performance depends on how to define the tabu list and parameter values, and (7) the definition of neighborhood is very important for all of the tested algorithms except GA.

Key Words: combinatorial optimization, metaheuristics, local search, GRASP, genetic algorithm, simulated annealing, tabu search, single machine scheduling.

1　序章

多くの組合せ最適化問題について, 厳密な最適解を求めることが極めて困難であることが, 計算の複雑さの理論により明らかにされてきた. NP 困難問題 [19] はその代表例である. しかし, 現実には, 最適性の保証はなくとも, 十分精度の高い解が求まれば満足のいく場合が多い. 近似解法 (approximate algorithms, 発見的手法 (heuristics)) はこのような目的に用いられる. 近似解法の基本戦略として, 欲張り法 (greedy method), 局所探索法 (local search, 以下 LS 法と記す) などがあるが, 最近では, これらを組合わせるなどの方法により, 多少時間はかかっても, より精度の高い解を求める枠組を提供しようとする, メタ戦略 (metaheuristics) の研究が盛んである. 代表的なものとして, ランダム多スタート局所探索法 (random multi-start local search, 以下 MLS 法と記す) [36, 44, 52], 遺伝アルゴリズム (genetic algorithm, 以下 GA 法と記す) [10, 25, 27, 38, 57], アニーリング法 (simulated annealing, 以下 SA 法と記す) [1, 35, 37], タブー探索法 (tabu search, 以下 TS 法と記す) [20, 21, 22, 23] などがある. これらの変形として, GA 法に LS 法を組み

込んだ遺伝的局所探索法 (genetic local search, 以下 GLS 法と記す) [5, 6, 18, 32, 39, 49, 56, 58], MLS 法の初期解生成ルーチンとして欲張り法を組合わせた GRASP 法 (greedy randomized adaptive search procedure) [13, 15, 16, 17, 41, 42, 54], MLS 法の初期解生成に過去の探索で得られた解を利用する反復局所探索法 (iterated local search, 以下 ILS 法と記す) [34, 46], SA 法を多少単純化した方法 (threshold accepting, 以下 TA 法と記す) [12, 43], TA 法の変形 (great deluge algorithm, 以下 GDA 法と記す) [11, 55] など, 様々な戦略が提案されている.

メタ戦略は, 非常に柔軟性に富む枠組みであるため, 内部に様々な工夫を施すことが可能であり, そのような工夫により, 強力なアルゴリズムを構成できる可能性を秘めている. 例えば, 行商人問題に対する Lin と Kernighan の方法 [44] は, MLS 法の一つであるが, 問題の構造をうまく利用した精巧な近傍を用いることにより成功をおさめた一例である. また, 解こうとしている問題の部分問題, あるいは緩和問題などに専用の近似, あるいは厳密解法が利用できる場合は, そのような手法を組合わせたハイブリッド・メタ戦略も可能である [30, 61].

メタ戦略のもう一つの魅力として, その簡単さとロバスト性が挙げられる. これらは, 問題の数学的構造などに対するとりわけ深い洞察がなくても, アルゴリズムを簡単に作ることができ, しかも, ある程度良い結果が期待できる. また, 単純な近似解法に比べ, より高い精度の解が得られる場合が多い. 本報告では, この点に重点を置き,「できるだけ単純な枠組みに従い, アルゴリズム内部のオペレータには複雑なものを用いない」ことに留意しつつ, 様々なタイプのメタ戦略について計算実験を行った結果について報告する ([59, 60] にも結果の一部を報告した). 本研究の目的は, 特定の問題に対する最良のアルゴリズムを見い出すことではなく, メタ戦略を「手軽なツール」として用いることを前提に, 問題固有の性質にできるだけ依存しないような一般的傾向を観察することにある. 現実の場面でメタ戦略のアルゴリズムを開発する際には, 開発にあまり時間をかけておれず, アルゴリズムの内部オペレータに複雑なテクニックを取り入れることは出来ない場合が多い. このような場面において, 本研究で観測されたデータは, アルゴリズム開発者の参考になり, 有用であると考えられる.

具体的な対象としては, 1 機械スケジューリング問題 (single machine scheduling problem, 以下 SMP と記す) [31] を用いた. SMP は, 与えられた n 個の仕事のコスト最小の順列を求める問題で, NP 困難であることが知られている.

$n = 100$ までの問題例に対する計算実験の結果, 以下のような傾向が観察できた. なお, MLS 法は, メタ戦略の中では最も単純であるが, 本研究で取り扱った SMP に対しては比較的良好な結果が得られたため, 他のメタ戦略の性能を評価する際の基準として用いた.

1. GRASP 法の性能は, 用いる欲張り法に大きく依存し, MLS 法に比べて性能が向上する場合もあれば, 劣る場合もある.
2. ILS 法の性能は, 用いる近傍によってやや傾向が異なるが, MLS 法に比べ有効である. さらに, 交換近傍 (4.2節で定義する) を用いた場合には, 調べた他のメタ戦略のいずれよりも高い性能が得られた.
3. GA 法は, 多くの場合, MLS 法と同等, あるいはそれ以下の性能を持つ. GLS 法は, 比較的長い計算時間が与えられた場合, MLS 法より有効である. また, GA および GLS 法の性能は, 内部オペレータの一つである交叉法に関してロバストである.
4. SA 法も, 比較的長い計算時間が許される場合は有効であり, その性能は, パラメータの値に大きくは依存しない. また, SA 法の変形である TA 法と GDA 法も, SA 法と同程度の性能を持つ.
5. TS 法の性能は, 用いるタブーリストやパラメータの値に大きく依存し, MLS 法に比べて性能が向上する場合もあれば, 劣る場合もある.
6. 近傍の定義は, GA 法を除く全ての戦略において極めて重要である.

なお, 本研究と動機は異なるが, メタ戦略のいくつかを比較した文献としては, [2, 8, 34, 55, 58] などがある. ただし, 比較実験による結果は, 取り扱う問題により異なる場合があるので, 注意する必要がある.

2 様々なメタ戦略

組合せ最適化問題に対する近似解法の基本戦略として,

- 欲張り法,
- 局所探索法 (LS 法),

などがある. 欲張り法は, 目的関数への貢献度を示す局所的な評価値に基づいて, 実行可能解を直接構成する方法である. LS 法は, 適当な候補解 x に対し, x に少しの変形を加えることによって得られる解集合 $N(x)$ (近傍と呼ばれる) 内に, x よりも良い解 x'があれば, $x := x'$と移動する操作を, 近傍内に改善解が存在しなくなるまで反復する方法である. $N(x)$ 内に改善解が存在しない x を局所最適解と呼ぶ. 一般に, LS 法を 1 度だけ用いることによって得られた解には, 高い精度はあまり期待できない. これは, 組合せ最適化問題においては, 解空間中に局所最適解が多数存在する傾向にあることが原因と考えられる. これを克服するため,

1. 複数個の初期解を試してみる,
2. 近傍を広くとるなど, 近傍を工夫する,
3. 改悪解への移動も許し, 移動のルールを工夫する,

などの方法が考えられる. これらの要素を取り込んだアルゴリズムの枠組をメタ戦略と呼んでいる. 代表的なメタ戦略としては,

- 多スタート局所探索法 (MLS 法),
- 遺伝アルゴリズム (GA 法),
- アニーリング法 (SA 法),
- タブー探索法 (TS 法),

などがあり, また, これらの変形や組合わせたものも提案されている. 以下, それぞれを簡単に説明する. なお, メタ戦略については [28, 29, 40, 51] に, メタ戦略を他の手法 (基本的な近似解法, メタ戦略同士, あるいは厳密解法など) と組合わせるハイブリッドなアプローチについては [30] に解説がある.

2.1 多スタート局所探索法

MLS 法は, 複数個の初期解に対して LS 法を適用し, 得られた最良の解を出力する方法で, メタ戦略の中では最も古くから広く用いられてきた手法の一つである [36, 44, 52]. 初期解は, ランダムに生成する, 欲張り法を利用する, などの方法が考えられる.

GRASP 法は, ランダム性を加えた欲張り法により生成された解を初期解とする方法で, Feo らのグループ (例えば [17]) により提案された. 欲張り法の各ステップでは, 通常, 目的関数への貢献度を示す局所的な評価値の最も高い要素を選択しつつ解を構成していくが, GRASP 法の初期解生成では, このとき, 評価値の高いものから順にいくつかを候補として持ち, それらの中からランダムに 1 つ選択を行うことにより, 解の生成にランダム性を加える. いくつかの組合せ最適化問題に対する適用結果 [13, 16, 41, 42, 54] にあるように, 一定の成果を収めている. 解説としては, [15] などがある. なお, LS 法との組合せは行われていないが, GRASP 法の初期解生成のメカニズムのアイディアは, [14, 26] などですでに紹介されている.

ILS 法では, 過去の探索で得られた良い解にランダムな変形を加えたものを初期解とする. ランダムな変形には, 通常, 近傍操作を用いるが, その際, LS 法に用いる近傍とはタイプの異なる近傍を利用する場合もある. ILS 法は, Johnson [34] により提案された. これをやや拡張した手法が [46] に紹介されている. また, 類似の方法で, noising method と呼ばれるものが, [7] で提案されている.

2.2 遺伝アルゴリズム

GA 法は, 生物の進化にアイディアを得た方法で, 複数の解を同時に保持し, それらを集団として改善していくところに特徴を持つ. 現在保持している解集合 P に対し, 交叉 (crossover) および突然変異 (mutation) の操作を加え, 新しい解集合 $Q \subset N(P)$ ($N(P)$ は交叉と突然変異によって生成され得る解全体の集合とする) を生成し, Pと Q から淘汰 (selection) の規則に従って解集合 $P' \subset P \cup Q$ を選択し, $P := P'$とする操作を反復するものである. 交叉は, 2 つまたはそれ以上の解を組合わせることにより新たな解を生成する操作, 突然変異は, 1 つの解に少しの変形を加えることで新たな解を生成する操作である. GA 法は, Holland [27] により提案された. 解説としては, [10, 25, 38, 57] などがある.

GLS 法は, GA 法の内部に LS 法を組合わせ, 交叉や突然変異によって生成された解を LS 法によって改善した後に淘汰を行う方法である. GLS 法の基本的なアイディアは [6] ですでに提案されているが, 比較的初期の研究としては [10, 25, 32, 33, 47, 48, 49, 56, 58] などがある. なお, genetic local search という用語を用いたのは, [58] が最初と思われる.

2.3 アニーリング法

SA 法は, LS 法の移動ルールに工夫を加えた方法の一つで, $N(x)$ 内の各解に, 解の良さに応じた遷移確率 (良い解ほど移行し易い) を設定し, それに従って次の解を選ぶ. 具体的には,

1. $N(x)$ 内の解をランダムに 1 つ選び x'とする.
2. x と x'の目的関数値の差を計算し, Δとする (x' が改悪解ならば$\Delta > 0$).
3. $\Delta \leq 0$ ならば 1, そうでなければ $e^{-\Delta/t}$の確率で $x := x'$とする.

このように, 改悪解であっても遷移する確率を与えることにより, 局所最適解からの脱出を図るものである. 遷移確率は, 物理現象の焼きなまし (annealing) にアイディアを借りて, 温度と呼ばれるパラメータ t により調整される. 温度 t は, 通常, 探索の初期の段階では高めに設定しておき, 探索が進むにつれて徐々に小さくしていく. $t \to 0$ では, SA 法の動作は通常の LS 法と同様になる. SA 法は Kirkpatrick ら [37] により提案された. 精密な計算実験の報告が [35] に, また, 解説としては [1] などがある.

TA 法は SA 法の変形の 1 つである. 具体的には, SA 法の反復のステップ 3 を,

3′. $\Delta < \tau$ならば $x := x'$とする,

で置き換える. τは探索を制御するパラメータで, 閾値 (threshold) と呼ばれる. 閾値τは, SA 法における温度 t と同様に, 探索の初期の段階では大きめに設定しておき, 探索が進むにつれて徐々に小さくしていく. SA 法では, ステップ 3 において, 乱数を 1 回生成し, 指数関数を 1 回計算する必要があるが, TA 法ではそれらが不要となる. それゆえ, 各反復での計算が多少効率化される. TA 法は, Dueck ら [12] により提案された. [2, 43, 58] などに, 他のメタ戦略との比較結果がある.

GDA 法は, TA 法の変形の 1 つである. 説明の都合上, 本節の GDA 法の説明に限って最大化問題を仮定する. 具体的には, SA 法の反復のステップ 2 と 3 を,

2′. x'の目的関数値が Wよりも大きければ, $x := x'$ とする,

で置き換える. Wは探索を制御するパラメータで, 水位 (water level) と呼ばれる. Wは, 通常, 探索の始めには十分小さい値にしておき, 反復毎に, 新たに採用された x の目的関数値と現在の Wの値の間の適当な値に更新することによって, 徐々に大きくしていく. Wを, 常に新たに採用された解の目的関数値に設定すれば, GDA 法の動作は LS 法と同様になる. 大洪水 (great deluge) が起こり, 水位が徐々に増してくる時に, 出来るだけ高い山に逃れたいが, 水位より下にある所には移動が出来ないという状況における人間の行動を想定してアルゴリズムが開発されたため, このような名前がつけられている. GDA 法は [11] で提案された. 他のメタ戦略との比較実験が [55] に報告されている.

2.4 タブー探索法

TS 法は, SA 法と同様, LS 法の移動ルールに工夫を加えたものであり, $N(x)\backslash(\{x\} \cup T)$ 中の最良の解を次の解として選ぶ. Tは, 解の循環を避けるために用意される, タブーリスト (tabu list, または短期メモリ) と呼ばれる解集合である. このように, TS 法では, 常に T以外の解に移行するため, Tに最近探索した解を含めておけば, T以外の新しい解への移動が強制され, 短い周期のサイクリングを防ぐことが出来る. TS 法では, さらに, 特定の変数を変更した頻度や, 探索してきた解の特徴を長期間に渡り記憶しておくこと (長期メモリと呼ばれる) により, 未探索の領域へ探索を方向づけようとする手法が組合わせて用いられることが多い. TS 法は Glover [20] により提案された. 解説としては [21, 22, 23] などがある.

3　1 機械スケジューリング問題

1 機械スケジューリング問題 (SMP) は, 代表的なスケジューリング問題の 1 つであり, これまで多くの研究がある [4, 31]. 与えられた n 個の仕事 $V = \{1, \ldots, n\}$ を 1 台の機械で処理するが, 機械は 1 度に 1 つの仕事しか処理できず, 処理の中断, 処理間の無駄時間は許されないと考える. 従って, スケジュールは n 個の仕事の順列σを与えることで決定される. $\sigma(k) = i$ (または$\sigma^{-1}(i) = k$) は, k番目に処理される仕事が i であることを意味する. 各仕事 i は処理時間 p_iを要し, 完了時刻が c_iであるときに $g_i(c_i)$ のコストがかかる. c_i は処理順序によって $c_i = \sum_{k=1}^{\sigma^{-1}(i)} p_{\sigma(k)}$と一意に定まる. このとき,

$$cost(\sigma) = \sum_{i \in V} g_i(c_i) \rightarrow \min$$

を実現するσを求める問題である. コストの一例として, 以下の実験では,

$$g_i(c_i) = h_i \max\{d_i - c_i, 0\} + w_i \max\{c_i - d_i, 0\}$$

を用いた. ここに $d_i \in Z_+$ (正整数の集合) は仕事 i の納期, $h_i, w_i \in Z_+$はそれぞれ仕事 i の進み, 遅れに対するペナルティの重みである. この定義に限らず, 多くの $g_i(c_i)$ について SMP の NP 困難性が知られている.

なお, この問題には, 本研究で取り扱う近傍操作 (4.2 節で定義する) に関連して, 以下のような性質がある. 交換近傍についてのみ説明するが, 挿入近傍についても同様である.

1. 近傍操作により, 交換された 2 つの仕事の間にスケジュールされている仕事については完了時刻 c_iが変化するが, それ以外の仕事の完了時刻は変化しない.
2. 局所最適解σの近傍内には, σと同じコストを持つ解が多数存在する. これは, 主に, 隣合う仕事の交換によってコストが変化しない場合があることが原因である.
3. 局所最適解σの近傍において, 良いコストを持つ解のほとんどは, 交換される仕事同士の位置が近い (即ち, 交換される仕事 i と jに対して $|\sigma^{-1}(i) - \sigma^{-1}(j)|$ が小さい).

ただし, 2 と 3 は, 予備実験により観測された性質である.

性質 2 と 3 より, 探索が局所最適解に到達した時, コストの増加量が少ない解に移動しようとすると, 隣同士, あるいは位置の近い仕事のペアが次の近傍操作として選ばれやすい. ところが, 性質 1 より, このような交換操作を何度か行っても, それらは互いに干渉しない場合が多い. その結果, 探索ルールの与え方によっては, サイクリングが起こりやすくなると考えられる.

4　計算実験

本節では, 計算に用いたアルゴリズムの詳細と計算結果について報告する. 計算実験は, Sun SPARC station IPX 上で行い, 言語は C 言語を用いた. 解の精度は, 全実験中に求まった最良解からの誤差で計る. また, アルゴリズムの効率は, 計算時間ではなく, コストの評価回数 (以下, サンプル数と記す) を基準とする. これは, 計算時間が, コンピューターやプログラム技術など, アルゴリズムの本質とは関係のない要因に左右されやすいことを考慮したためである.

以下では, GRASP 法以外では初期解はランダムに生成し, 近傍 $N(\sigma)$ 内の探索順序は前もって定められたランダムな順序に従うものとする.

4.1　問題例の生成

問題例は, $n = 35, 100$ のそれぞれに対し 10 問ずつ, 以下のようにランダムに生成した. これは, SSDP 法 [31] によって厳密解の得られる問題例 (最大 $n = 35$) と, 比較的サイズの大きい問題例 ($n = 100$) を調べるためである. 各 i に対し, まず p_i, h_i, w_i $(i \in V)$ を区間 $[1, 10]$ の整数の中からランダムに選ぶ. 問題の難しさは RDD (relative range of due date) と LF (average lateness factor) という 2 つのパラメータに関係するということが観測されている (例えば [53]). ここでは

$$RDD = 0.2, 0.4, 0.6, 0.8, 1.0$$
$$LF = 0.2, 0.4$$

とした. これら $5 \times 2 = 10$ の場合それぞれに対応して納期 d_i $(i \in V)$ を区間

$$[(1 - LF - RDD/2)T, (1 - LF + RDD/2)T]$$

の整数の中からランダムに選んだ.

4.2 ランダム多スタート局所探索法

LS 法や MLS 法の枠組みの中でも, (1) 近傍の定義, (2) 移動ルール (近傍内をどのような順序で探索し, どの解に移動するか) について, 様々な工夫を加えることが可能である. ここでは, できるだけ単純な操作のみを用いてアルゴリズムを構成することに留意し, 以下の組合せを調べる.

(1) 近傍は, $N_{ins}(\sigma) = \{\sigma$の 1 つの仕事を他の位置に挿入することにより得られる解 $\}$ (挿入近傍と呼ぶ) と, $N_{swap}(\sigma) = \{\sigma$の 2 つの仕事の位置を交換することにより得られる解 $\}$ (交換近傍と呼ぶ) の 2 通り.

(2) 移動ルールは, 近傍内をランダムな順序で調べ, 最初に見つかった改善解に移動する方法 (FIRST) と, 近傍内の最良解に移行する方法 (BEST) の 2 通り.

表 1に, 上述の近傍と移動ルールの組合せについて, $n = 35$ ではサンプル数 30 万, $n = 100$ では 300 万で探索を打ち切った時の, MLS 法による解の精度 (実験中得られた最良の解のコスト値との相対誤差 (%)) の 10 問に対する平均値を示す. また, 括弧内に, スタート回数 (調べた初期解の個数) の平均値を併記する.

表 1: MLS 法による最良解の平均誤差 (%). 括弧内の数値はスタート回数の平均値.

	$n = 35$; 30 万サンプル		$n = 100$; 300 万サンプル	
	N_{ins}	N_{swap}	N_{ins}	N_{swap}
FIRST	0.000	0.000	0.669	0.182
	(97.6)	(131.0)	(103.6)	(126.5)
BEST	0.289	0.047	4.696	0.624
	(8.6)	(13.9)	(3.2)	(5.0)

表より, (1) 移動ルールに FIRST を用いる方が, BEST を用いた場合に比べて, より良い解が速く求まる, (2) 近傍に N_{swap}を用いる方が N_{ins}を用いた場合よりもやや精度が高い, ことが観測できる. この結果をふまえ, 以下の実験では移動ルールを FIRST に限定する.

なお, 移動ルールについては, スタート回数をともに 100 回ずつとし, 得られた局所最適解の精度の平均値を比較した結果, FIRST と BEST の間に大差はないことが観測されている [59]. このように, どちらの方法を用いても, 得られる個々の局所最適解の精度は平均的にはあまり変わらないが, 同じサンプル数で探索を打ち切って比較を行う場合には, FIRST を用いる方が列挙できる局所最適解の数が多いことが, 上記 (1) の原因と考えられる.

4.3 GRASP 法

初期解生成には以下のような欲張り法を用いる. 各ステップにおいて, スケジュールが未定である仕事の集合を M $(\subseteq V)$ とする. 処理順序の前から順に, Mの中で局所利得 $e(i)$ (例えば, 仕事 i を次の仕事とし, 残りの $M \backslash \{i\}$ の仕事を納期の小さい順に並べた時のコスト) を最大にする仕事 i を次の仕事とする操作を, スケジュールが完了するまで繰り返す. 局所利得 $e(\cdot)$ を工夫することで, 様々な欲張り法が構成できる. これにランダム性を加えるため, 各ステップにおいて, $e(i)$ の大きい方から定数個を集めた集合 $CA \subseteq M$ ($|CA|$ はアルゴリズムのパラメータ) から 1 つの仕事をランダムに選び, 次の仕事とする. これと対称的に, 後から順にスケジュールを行うことも可能である. 紙面の都合上, 説明は略すが, 6 通りの局所利得関数 [59] のそれぞれに対して前向きと後向きの 2 通りのスケジュールを考えることで, 合計 12 通りの欲張り法を調べる.

12 通りの局所利得関数のそれぞれに対し, パラメータを $|CA| = 1, 2, 4, 7, 10, 20$ と変化させたときの GRASP 法の性能を調べた. $|CA| = 1$ は, ランダム性を加えない基本的な欲張り法を意味する. 表 2に, $n = 100$ の問題例に対し, サンプル数 300 万で探索を打ち切ったときの解の精度 (%) を示す. 近傍に N_{swap} を用いた場合の結果のみ示すが, N_{ins}についても同様の結果が得られている. 表中, $e_i(\mathrm{f})$ (前向き) と$e_i(\mathrm{b})$ (後向き) は, 様々な局所利得関数を, init は GRASP 法の初期解, 即ち, 欲張り法 ($|CA| = 1$) による解の精

度を表す. 比較のため, MLS 法による解の精度も併記する.

表 2: $n = 100$ の問題例に対し, サンプル数 300 万で探索を打ち切ったときの, N_{swap}を用いた GRASP 法による最良解の平均誤差 (%). 最終列 init は, 欲張り法による初期解の平均誤差.

$\lvert CA\rvert$	1	2	4	7	10	20	init
e_1(f)	0.183	0.156	0.228	0.170	0.169	0.151	50.5
e_1(b)	0.205	0.309	0.193	0.223	0.230	0.154	53.8
e_2(f)	1.524	1.292	1.034	0.653	0.446	0.214	28.6
e_2(b)	1.339	1.117	0.650	0.423	0.270	0.150	33.6
e_3(f)	0.383	0.335	0.202	0.136	0.077	0.163	8.8
e_3(b)	0.568	0.398	0.193	0.220	0.137	0.183	16.0
e_4(f)	0.170	0.099	0.121	0.097	0.153	0.189	18.2
e_4(b)	0.466	0.392	0.214	0.308	0.205	0.241	72.7
e_5(f)	0.120	0.171	0.117	0.147	0.069	0.145	73.2
e_5(b)	0.138	0.109	0.122	0.119	0.095	0.131	90.6
e_6(f)	0.172	0.135	0.053	0.106	0.109	0.107	116.0
e_6(b)	0.158	0.123	0.129	0.135	0.088	0.138	64.3
MLS			0.182				314.6

表より, (1) GRASP 法の性能は, 初期解生成に用いる局所利得関数に大きく依存する, (2) 局所利得関数をうまく選べば, パラメータ $|CA|$ に大きく左右されることなく MLS 法よりも高い性能が得られる, (3) 良い初期解が GRASP 法の性能向上に直結するとは限らない, ことが観測できる.

4.4 反復局所探索法

ILS 法の枠組みの中では, (1) 初期解生成に利用する解の選択方法, (2) 初期解生成の際に加えるランダムな変形に利用する操作 (以下では突然変異と記す) について, 様々な工夫を加えることが可能である. (1) については, ILS 法では通常, 複数個の解を保持することは行わず, 過去の探索における最良解 (σ_{best}と記す) と, 前回の反復における局所最適解 (σ_{last}と記す) のいずれかを利用する. ここでは, [46] の流儀に従い, 以下の方法をとる.

(1) $\Delta := cost(\sigma_{last}) - cost(\sigma_{best})$ とし, $\Delta \leq 0$ ならば確率 1 で, $\Delta > 0$ ならば確率 $e^{-\Delta/t}$でσ_{last}を選択する. そして, σ_{last}が選ばれなかったときにはσ_{best}を選択する. ただし, t は固定されたパラメータとし, SA 法のような制御は行わない.
また, (2) については, できるだけ単純な操作のみを用いることに留意し, 以下の方法をとる.

(2) 突然変異は, 解σに対し, 近傍 $N(\sigma)$ 中から解σ'をランダムに 1 つ選び, $\sigma := \sigma'$とする操作とする. 近傍 Nには, N_{ins}と N_{swap}の 2 通りを考え, それぞれ INS, SWAP と記す.

表 3に, $n = 100$ の問題例に対し, 突然変異に INS と SWAP のそれぞれを用いた場合について, パラメータを $t = 0, 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, \infty$ と変化させたときの ILS 法の解の精度 (%) を示す. サンプル数は 300 万とした. なお, $t = 0$ は, 初期解生成に用いる解として, 常にσ_{best}を, $t = \infty$ はσ_{last}を選ぶことを意味する. $n = 35$ の問題例に対しては, 試みたほとんど全ての場合において, 最適解が得られたため, 結果は省略する. 比較のため, MLS 法による解の精度も併記する.

表 3: $n = 100$ の問題例に対し, サンプル数 300 万で探索を打ち切ったときの, ILS 法による最良解の平均誤差 (%).

	N_{ins}		N_{swap}	
t	INS	SWAP	INS	SWAP
0	0.312	0.360	0.008	0.001
1	0.424	0.401	0.015	0.024
2	0.362	0.570	0.012	0.023
4	0.523	0.334	0.025	0.001
8	0.422	0.205	0.012	0.005
16	0.668	0.272	0.009	0.027
32	0.726	0.660	0.023	0.009
64	0.830	0.527	0.054	0.011
128	0.892	0.689	0.046	0.060
256	1.136	0.661	0.075	0.032
512	0.936	1.064	0.125	0.056
1024	0.926	1.158	0.138	0.064
∞	1.211	1.249	0.105	0.061
MLS	0.669		0.182	

表より, (1) ILS 法は, MLS 法に比べ, 精度の高い解を得ることが出来る, (2) 近傍に N_{swap}を用いた方が, N_{ins}を用いた場合に比べ効果が大きい, (3) ILS 法の性能は, 突然変異にはあまり大きくは左右されない, (4) パラメータ t の値は小さい方が良い, ことが観測できる.

4.5 遺伝アルゴリズム

交叉, 突然変異, および淘汰の規則には, 様々なものが考えられるが, ここでは, 以下を試みた.

交叉: order crossover [9, 49] を用いる. 交叉の基となる親の数と, 交叉によって生成される子の数の組合せはいろいろ考えられるが, ここでは 2 つの親σ_Aとσ_Bから 1 つの子σ_Cが作られるものとする.

まず, n ビットのマスク $m \in \{0,1\}^n$をランダムに発生し, 各位置 kについて $m(k) = 0$ ならば$\sigma_C(k) := \sigma_A(k)$ とする. $D_B := \{(i,j) \mid \sigma_B^{-1}(i) \leq \sigma_B^{-1}(j)\}$, $S_m := \{\sigma_A(k) \mid m(k) = 0\}$ と定義すると, $D'_B := D_B \backslash \{(i,j) \mid i \in S_m$ または $j \in S_m\}$ は$V \backslash S_m$の全順序である. $m(k) = 1$ の各位置に, 前から順に D'_Bの順序に従って$V \backslash S_m$の要素を割り当てていくことにより, 子σ_Cが完成する.

このように, ランダムに発生したマスクに従ってどちらの親から要素を継承するかを決定する手法を一般に一様交叉 (uniform crossover) と呼ぶ. 特に高々l 点で 0 と 1 が隣接するマスクのみを発生する場合を, l点交叉という. 例えば, マスク 11000, 10011 はそれぞれ 1 点交叉, 2 点交叉に対応する. 以下では, 1 点, 2 点, 一様交叉を考え, それぞれ OX1, OX2, OXU と記す.

他の交叉法 (例えば, partially mapped crossover [24] や cycle crossover [50] など) についても実験を行ったが, 紙面の都合上, 結果は省略する. これらの交叉法を含めた比較については, [59] を参照のこと.

突然変異: 突然変異は, ILS 法と同様に, INS と SWAP の 2 通りを調べる.

淘汰: ここでは, 簡単のため, 以下のように子を 1 つ生成する毎に淘汰を行う. (つまり, 1 世代に生成される子の数$|Q|$を 1 と考える.) 2 つの親σ_Aとσ_Bをランダムに選び, そのいずれかに突然変異を加え, その後σ_Aとσ_Bを交叉させることにより, 子σ_Cを生成する. そして, 全ての$\sigma \in P \cup Q$ に対して $cost(\sigma_{worst}) \geq cost(\sigma)$ をみたす解$\sigma_{worst} \in P \cup Q$ に対し, $P := P \cup \{\sigma_C\} \backslash \{\sigma_{worst}\}$ として入れ換える. なお, 解集合 Pには重複解を許さないものとし, 子を生成した時点で$\sigma_C \in P$であるときは, 解の入れ換えを行わない.

$|P| = 100$ とし, 上述の交叉, 突然変異, および淘汰の様々な組合せを比較した. なお, $|Q| = 1$ としたため, 1 世代におけるサンプル数 (即ちコストの評価回数) は 1 回となる. 表 4に, $n = 35$ ではサンプル数 3 万, $n = 100$ では 30 万で探索を打ち切ったときの解の精度 (%) を示す. 表中, 'MUT only' は交叉を用いなかったことを, 'no MUT' は突然変異を用いなかったことを意味する. 比較のため, MLS 法による解の精度も併記する. ただし, MLS 法の近傍には, 突然変異と同じ操作を用いる (例えば, INS の欄には N_{ins}を用いた MLS の結果を示す).

表 4: $|P| = 100$ とした GA 法による最良解の平均誤差 (%).

	$n = 35$; 3 万サンプル			$n = 100$; 30 万サンプル		
	no MUT	INS	SWAP	no MUT	INS	SWAP
OX1	110.1	5.1	3.5	190.9	3.1	2.1
OX2	10.5	0.2	0.8	35.9	2.6	0.7
OXU	1.4	0.3	1.1	3.8	0.8	0.8
MUT only	—	2.3	4.0	—	1.4	1.5
MLS	—	0.1	0.06	—	1.0	0.5

表より, 以下のことが確認できる. (1) GA 法により良質の解を得るためには, 突然変異は不可欠である. (2) 突然変異を用いない場合, GA 法の性能は交叉法に大きく依存してしまうが, 突然変異を用いる場合には, 交叉法にはほとんど依存しない. (3) 交叉を用いた GA 法に比べ, MUT only では同定度の精度の解を得るのにサンプル数がやや多く必要であることから, 交叉も GA 法の性能向上に寄与していると言える. (4) MLS 法の方が GA 法に比べ性能が高い.

次に, 表 5に, $|P|$ を $10^1 \sim 10^4$と変化させたときの結果を示す. $n = 100$ の問題例に対し, サンプル数 300 万で探索を打ち切ったときの解の精度 (%) である. 交叉には OXU を用いた.

表 5: $n = 100$ の問題例に対し, 300 万サンプルで探索を打ち切った時の, GA 法による最良解の平均誤差 (%).

$\|P\|$	10^1	10^2	10^3	10^4	MLS
INS	4.876	0.641	0.595	0.524	0.669
SWAP	0.804	0.802	0.538	0.327	0.182

表より, 以下が確認できる. (1) $|P|$ が大きいほど, 解の精度は高くなる. (2) $|P|$ を大きくしてもなお, GA 法による解の精度は, MLS 法と同等, あるいはそ

れ以下である.

なお, ここで用いた GA 法では, MLS 法などの他のメタ戦略に比べ, 同数の解をサンプルするのにかかる計算時間が非常に大きくなってしまう. これは, (1) 近傍操作に比べ, 交叉の操作がやや複雑であること, (2) 解集合 Pの中に重複解がないかどうかを調べるのに時間がかかること, が主な原因である. また, 一般に, 近傍操作や交叉の操作に伴うコスト評価の計算は, 交叉の方が効率が悪い場合が多いと考えられる. これらのことを考慮に入れると, 単純な GA 法は, 組合せ最適化問題に対する近似解法としては, あまり性能は高くないといえる.

4.6 遺伝的局所探索法

GLS 法では, 交叉や突然変異によって生成された Q の解を, LS 法により局所最適解まで改善した後, 淘汰を行う. 交叉, 突然変異, および淘汰の規則は, GA 法と同様とする.

表 6に, $n = 100$ の問題例に対し, $|P| = 20$ とし, サンプル数 300 万で探索を打ち切ったときの, GLS 法による解の精度 (%) を示す. $n = 35$ の問題例に対しては, 試みたほとんど全ての場合において最適解が得られたため, 結果を省略する. 参考のため, MLS 法による結果も併記する.

表 6: $n = 100$ の問題例に対し, $|P| = 20$ とし, サンプル数 300 万で探索を打ち切った時の, GLS 法による最良解の平均誤差 (%).

neighbor	N_{ins}			N_{swap}		
mutation	noMUT	INS	SWAP	noMUT	INS	SWAP
OX1	0.281	0.361	0.303	0.090	0.065	0.069
OX2	0.220	0.327	0.272	0.038	0.052	0.047
OXU	0.203	0.164	0.266	0.015	0.069	0.038
MUT only	—	0.598	0.510	—	0.103	0.057
MLS		0.669			0.182	

表より, 以下が観測できる. (1) GLS 法は, GA 法や MLS 法に比べ, より精度の高い解を得ることができる. (なお, この傾向は, 例えば $n = 100$ の問題例に対してサンプル数 30 万程度の短い計算時間しか許されないときには現れず, サンプル数 300 万程度の, 比較的長い計算時間が許されるときに現れることが予備実験により確認されている.) (2) GLS 法の性能は, 用いる交叉や突然変異に大きくは依存しない. 交叉と突然変異を比べると, 交叉の方がやや効果が大きい. 一方, 通常は近傍操作をそのまま突然変異として利用できるため, 突然変異のみを用いた GLS 法の方が, 開発は容易である. (3) 解の精度は用いる近傍に大きく依存する.

なお, ILS 法 (パラメータ $t = 0$ とした場合) は, GLS 法において, $|P| = 1$ とし, 交叉を用いない特殊ケースとみなすことができる. 表 3と表 6を比較することにより, 以下が確認できる: 交換近傍を用いた場合は ILS 法の方が, 挿入近傍を用いた場合は GLS 法の方が良い成果をあげているが, 両者の性能に大差はない.

4.7 アニーリング法

SA 法には, [35] と同様の枠組みを用いた. パラメータとして, IP, TR, SF, TFの 4 つを用意する. まず, 初期温度 t は, ランダムに生成した初期解σに対し,

$$\left(\sum_{\sigma' \in UP(\sigma)} e^{-\{cost(\sigma') - cost(\sigma)\}/t} \right) / |UP(\sigma)| \simeq IP$$

となるように定める. ここに,

$$UP(\sigma) := \{\sigma' \in N(\sigma) \mid cost(\sigma') > cost(\sigma)\}$$

である. 次に, kの初期値を 0 として, 以下のループを反復する. なお, kは, 改善解への移動が起こらないループが連続する回数を数えるカウンタである.

$k < TR \cdot |N|$ が成立している間, 以下を行う.

(a) 以下のループを $SF \cdot |N|$ 回反復する.

i. $N(\sigma)$ 中よりランダムに 1 つ解を選び, σ'とする.

ii. $\Delta := cost(\sigma') - cost(\sigma)$ とする.

iii. $\Delta \leq 0$ ならば確率 1 で, $\Delta > 0$ ならば確率 $e^{-\Delta/t}$で, $\sigma := \sigma'$とする.

iv. $\Delta < 0$ ならば $k := 0$, $\Delta \geq 0$ ならば $k := k + 1$ とする.

(b) $t := TF \cdot t$ とする.

上記の反復を終了した時点で, サンプル数が指定した数値に達していない場合には, ランダムな初期解より再び反復を行う.

パラメータ TRは, 予備実験により, N_{ins}を用いた場合は 1, N_{swap}を用いた場合は 2 とし, パラメータ TFは, 文献 [35] で推奨値となっていた 0.95 に固定した.

まず, $SF = 1$ として, パラメータ IPの効果を調べたところ, 以下の傾向が観測できた. (1) 得られた解の精度は, $IP \leq 0.1$ までは IPが大きくなるほど良くなるが, それ以上の値にしてもあまり変化はみられない. (2) 1 回の反復 (ランダムな初期解から始め, 上記のループを終了するまで) に要するサンプル数は, IPとともに増大する.

次に, $IP = 0.3$ として, パラメータ SFの効果を調べたところ, 以下の傾向が観測できた. (1) 得られた解の精度は, $SF \leq 1$ までは SFが大きくなるほど良くなるが, それ以上の値にしてもあまり変化はみられない. (2) 1 回の反復に要するサンプル数は, SFとともに増大する.

以上の結果より, $IP = 0.1, 0.3$, $SF = 1, 2$ の 4 通りの組合せを調べた. 表 7に, $n = 35$ ではサンプル数 30 万, $n = 100$ では 300 万で探索を打ち切ったときの解の精度 (%) を示す. 参考のため, MLS 法の結果も併記する.

表 7: SA 法による最良解の平均誤差 (%).

		$n = 35$; 30 万サンプル		$n = 100$; 300 万サンプル	
IP	SF	N_{ins}	N_{swap}	N_{ins}	N_{swap}
0.1	1	0.144	0.000	0.184	0.034
0.1	2	0.124	0.003	0.217	0.032
0.3	1	0.035	0.000	0.153	0.029
0.3	2	0.033	0.000	0.227	0.014
MLS		0.000	0.000	0.669	0.182

表より, 以下が観測できる. (1) SA 法は, 比較的長い計算時間が, 許されれば, MLS よりも精度の高い解を得ることができる. また, この効果が現れるのは, n が比較的大きい場合である. (2) SA 法により得られる解の精度は, パラメータにあまり大きくは依存しない. (一方, 1 回の反復に要するサンプル数は, パラメータに大きく依存する.) (3) 得られる解の精度は, 用いる近傍に大きく依存する.

4.8 TA 法と GDA 法

ここで用いる TA 法は, 前節の SA 法と同様に, IP, TR, SF, TFの 4 つのパラメータを用意する. まず, 初期閾値τは, ランダムに生成した初期解σに対し,

$$\{\sigma' \in UP(\sigma) \mid cost(\sigma') - cost(\sigma) \leq \tau\} / |UP(\sigma)| \simeq IP$$

となるように定める. 次に, SA 法のステップ (a) の iii を,

iii′. $\Delta < \tau$ならば$\sigma := \sigma'$とする.

で置き換え, さらにステップ (b) の t をτに置き換えて, 同様のループを反復する.

パラメータは, SA 法と同様に, まず, $TF = 0.95$ とし, さらに, 予備実験により, TRを, N_{ins}を用いる場合は $TR = 1$, N_{swap}を用いる場合は $TR = 2$ と定めた. 次に, IPと SFについて SA 法と同様の実験を行った. その結果, SA 法に類似の傾向はあるものの, SA 法ほどはっきりとした傾向は観測できなかった.

以上の結果より, TA 法のパラメータは, $IP = 0.1, 0.3$, $SF = 0.25, 0.5, 1, 2, 4$ の 10 通りを調べた. 表 8に, $n = 35$ ではサンプル数 30 万, $n = 100$ では 300 万で探索を打ち切ったときの解の精度 (%) を示す. 参考のため, MLS 法の結果も併記する. 表より, TA 法は, SA 法と同程度の性能を持つことが確認できる.

表 8: TA 法による最良解の平均誤差 (%).

		$n = 35$; 30 万サンプル		$n = 100$; 300 万サンプル	
IP	SF	N_{ins}	N_{swap}	N_{ins}	N_{swap}
0.1	0.25	0.003	0.000	0.093	0.031
0.1	0.5	0.000	0.000	0.155	0.037
0.1	1	0.243	0.003	0.196	0.006
0.1	2	0.376	0.000	0.326	0.041
0.1	4	0.364	0.132	0.385	0.019
0.3	0.25	0.000	0.000	0.130	0.031
0.3	0.5	0.101	0.018	0.165	0.031
0.3	1	0.082	0.023	0.200	0.039
0.3	2	0.146	0.023	0.401	0.029
0.3	4	0.246	0.023	0.336	0.040
MLS		0.000	0.000	0.669	0.182

GDA 法には, 以下の枠組みを用いた. パラメータ RS (rain speed と呼ばれる) と TRを用意する. 次に,

ランダムに生成した初期解σに対し, 初期水位 $W := cost(\sigma)$ とし, さらにkの初期値を 0 として以下のループを反復する.

$k < TR \cdot |N|$ が成立している間, 以下を行う.

(a) $N(\sigma)$ 中よりランダムに 1 つ解を選び, σ'とする.

(b) $cost(\sigma') < W$ならば $\sigma := \sigma'$とし, さらに $W := W - RS(W - cost(\sigma'))$ とする.

(c) $cost(\sigma') - cost(\sigma) < 0$ ならば $k := 0$, さもなければ $k := k + 1$ とする.

TRは, 予備実験により, 近傍に N_{ins}を用いる場合は 1, N_{swap}を用いる場合は 2 とした. また, $RS = 0.00125 \sim 0.64$ と変化させて傾向を調べたところ, (1) 得られる解の精度は RSが小さいほど高い, (2) 1 回の反復に必要なサンプル数は, RSが小さくなるにつれ増大する, ことが観測された.

表 9に, $RS = 0.005, 0.01$ のそれぞれについて, $n = 35$ ではサンプル数 30 万, $n = 100$ では 300 万で探索を打ち切ったときの GDA 法による解の精度 (%) を示す. 表より, GDA 法は SA 法と同程度の性能を持つことが確認できる.

表 9: GDA 法による最良解の平均誤差 (%).

	$n = 35$; 30 万サンプル		$n = 100$; 300 万サンプル	
RS	N_{ins}	N_{swap}	N_{ins}	N_{swap}
0.005	0.069	0.228	0.239	0.048
0.01	0.000	0.015	0.227	0.027
MLS	0.000	0.000	0.669	0.182

なお, 厳密な最適解への漸近収束性が, ある仮定の下で成立することが, SA 法については知られており (例えば [45]), それが SA 法の魅力の一つととらえられている. しかし, TA 法についても (SA 法ほど強い結果ではないとはいえ) 漸近収束に関する同様の結果がある [3]. また, 漸近収束性はあくまでも「十分に計算時間をかけた」時の理論的性質であり, 実際に用いられるアルゴリズムの性能に直接関係するとはあまり考えられない. SA 法の変形である TA 法や GDA 法によっても比較的良好な結果が得られていることを考えると, 物理現象のアナロジーや理論的成果にあまりとらわれず, 柔軟な姿勢でアルゴリズムを構築していくことが重要であると思われる.

4.9 タブー探索法

ここで扱う TS 法では, $N(\sigma)$ の解を順番に調べて行き, $cost(\sigma') < cost(\sigma)$ かつ $\sigma' \notin \{\sigma\} \cup T$, または, $cost(\sigma') < bestcost$ (aspiration criteria) をみたす最初の解σ'に移動する. Tは移動が禁止される解集合 (タブーリスト), $bestcost$ は過去の探索中に見つかった最良解である. このルールにより選ばれる $N(\sigma)$ 内の解がない場合 (即ち, σは (Tによって制限された近傍における) 局所最適解) は, 以下のルールに従って次の解を選ぶ.

カウンタ MCとパラメータ Rを用意し, MCの初期値を 0 とする. Rは局所最適解からの移動を行う際に長期メモリを使う頻度を制御するパラメータ (定数), MCは長期メモリを用いない移動が連続する回数を数えるカウンタである.

1. $cost(\sigma) < bestcost$ ならば, $MC := 0$ とする.
2. $MC < R$ならば, $cost(\sigma')$ を最小にする$\sigma' \in N(\sigma)\backslash(\{\sigma\} \cup T)$ を選び, $MC := MC + 1$ とする.
3. $MC \geq R$ならば, $cost(\sigma') + \alpha \cdot penalty(\sigma')$ を最小にする$\sigma' \in N(\sigma)\backslash(\{\sigma\} \cup T)$ を選び, $MC := 0$ とする.

αはアルゴリズムのパラメータ, $penalty(\sigma')$ は長期メモリのペナルティである.

タブーリストは T_{job}と T_{pos}の 2 つを試みた. これらは, $T_{job}(\sigma) = \{\sigma' \in N(\sigma) \mid \sigma$から$\sigma'$への移動は, 過去 TT回の移動において位置が変わった仕事の位置を再び変更する.$\}$, $T_{pos}(\sigma) = \{\sigma' \in N(\sigma) \mid \sigma$から$\sigma'$への移動は, 過去 TT回の移動の間に位置が変わった仕事をもとの位置にもどす.$\}$ と定義される. ただし, TTはタブー期間 (tabu tenure) と呼ばれるパラメータである.

長期メモリのペナルティには, $penalty_{move}$と $penalty_{period}$の 2 つを試みた. $LT_{move}(i, k)$ を, 過去の探索において, 仕事 i が位置 kから他へ移った移動の回数, $LT_{period}(i, k)$ を, 仕事 i が位置 kにスケジュールされていた期間の長さ (移動回数で計る) とする. そし

て, $penalty_{move}(\sigma') = \sum_{j \in CH(\sigma')} LT_{move}(j, \sigma^{-1}(j))$, $penalty_{period}(\sigma') = \sum_{k=1}^{n} LT_{period}(\sigma'(k), k)$ と定義する. ただし, $CH(\sigma')$ は, σからσ'への移動により位置が変更される仕事の集合である.

まず, 短期メモリの効果を調べるため, αを 0 として実験を行った. 2 タイプのタブーリスト T_{job}と T_{pos} について, TTの値を変化させたときの様子を調べた結果, 以下のことが観測された. (1) 近傍に N_{swap}, タブーリストに T_{pos}を用い, $TT = 1 \sim 5$ とした TS 法では, MLS 法よりも精度の高い解が得られた. (2) 近傍とタブーリストに他の組合せを用いた TS 法では, MLS 法よりも良い性能は得られなかった. この結果より, 以下の実験では, タブーリストには T_{pos}を用いる.

次に, 長期メモリの効果を調べる. 2 タイプのペナルティ$penalty_{move}$と $penalty_{period}$について, パラメータαを変化させたときの様子を調べた. その結果, 以下のことが観測された. (1) 近傍に N_{swap}, 長期メモリに $penalty_{period}$を用い, $\alpha \geq 10^4$とした TS 法では, MLS 法よりも高い精度の解が得られた. (2) 近傍と長期メモリに他の組合せを用いた TS 法では, MLS 法よりも良い性能は得られなかった. この結果より, 以下の実験では, 長期メモリには $penalty_{period}$を用いる.

最後に, 上記の実験によって選ばれたタブーリストと長期メモリを用いた TS 法の性能を評価する. パラメータαは, $n = 35$ では 10^3, $n = 100$ では 10^4とする. $R = TT$とし, $TT = 1, 3, 5, 7, 10, 20$ を調べた. 表 10に, サンプル数を $n = 35$ では 30 万, $n = 100$ では 300 万で探索を打ち切ったときの, TS 法による最良解の平均誤差 (%) を示す.

表 10: TS 法による最良解の平均誤差 (%).

	$n = 35$; 30 万サンプル		$n = 100$; 300 万サンプル	
TT	N_{ins}	N_{swap}	N_{ins}	N_{swap}
1	0.000	0.000	1.449	0.086
3	0.037	0.000	0.965	0.150
5	0.084	0.000	1.647	0.093
7	0.358	0.000	1.689	0.167
10	0.360	0.000	1.684	0.148
20	0.542	0.000	1.899	0.204
MLS	0.000	0.000	0.669	0.182

表より, 以下が観測できる. (1) 近傍に N_{swap}を用いた場合には, TS 法は MLS 法よりも高い精度の解を得ることができ, その性能は, パラメータ TTに大きくは依存しない. (2) 近傍に N_{ins}を用いた場合には, TS 法の性能は, MLS 法にやや劣る. (3) TS 法の性能は, 用いるメモリ (タブーリストや長期メモリ) のタイプに大きく依存する.

5 メタ戦略の比較

本節では, 様々なメタ戦略の比較を行う. 図 1と 2に, $n = 100$ の問題例に対して, 各メタ戦略で得られた最良解の平均誤差 (%) の, サンプル数に対する変化の様子を示す. 挿入近傍を用いた場合の結果が図 1, 交換近傍を用いた場合の結果が図 2である. GA 法については, N_{ins}と N_{swap}に対応して, 突然変異にそれぞれ, INS と SWAP を用いた場合の結果を示す. 各アルゴリズムのパラメータ設定等は, 以下の通りである.

GRASP 法: 近傍に N_{ins}を用いた場合には, 局所利得関数に $e_6(\mathrm{b})$ を用い, $|CA| = 7$ とし, N_{swap}を用いた場合には, 局所利得関数に $e_6(\mathrm{f})$ を用い, $|CA| = 4$ とする.

ILS 法: 突然変異には, SWAP を用い, 近傍に N_{ins}を用いた場合には$t = 8$, N_{swap}を用いた場合には $t = 0$ とする.

GA 法: 交叉には OXU を用い, $|P| = 1000$ とする.

GLS 法: 交叉には OXU を用い, $|P| = 20$ とする. また, 突然変異は, 近傍に N_{ins}を用いた場合には INS を用い, N_{swap}を用いた場合には用いない.

SA 法: 近傍に N_{ins}を用いた場合には, $IP = 0.3$, $TR = 1$, $SF = 1$, $TF = 0.95$ とし, N_{swap}を用いた場合には, $IP = 0.3$, $TR = 2$, $SF = 2$, $TF = 0.95$ とする.

TA 法: 近傍に N_{ins}を用いた場合には, $IP = 0.1$, $TR = 1$, $SF = 0.25$, $TF = 0.95$ とし, N_{swap}を用いた場合には, $IP = 0.1$, $TR = 2$, $SF = 1$, $TF = 0.95$ とする.

GDA 法: $RS = 0.01$ とする.

TS 法: タブーリストには T_{pos}, 長期メモリには $penalty_{period}$を用い, 近傍に N_{ins}を用いた場合には, $R = 3$, $TT = 3$, $\alpha = 10^4$とし, N_{swap}を用いた場合には, $R = 5$, $TT = 5$, $\alpha = 10^4$とする.

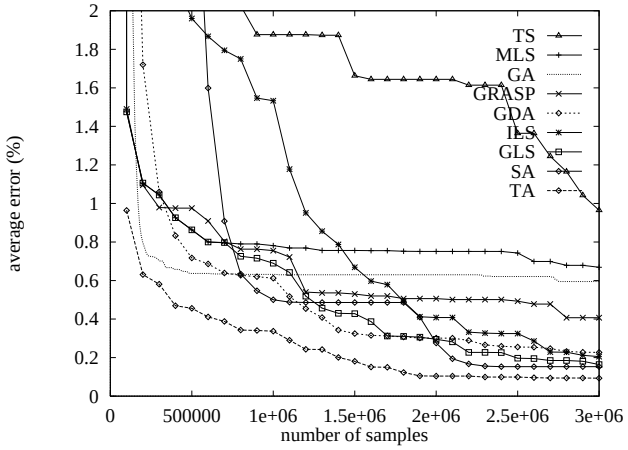

図 1: サンプル数に対する解の精度 (%) (近傍 N_{ins}, 突然変異 INS の場合).

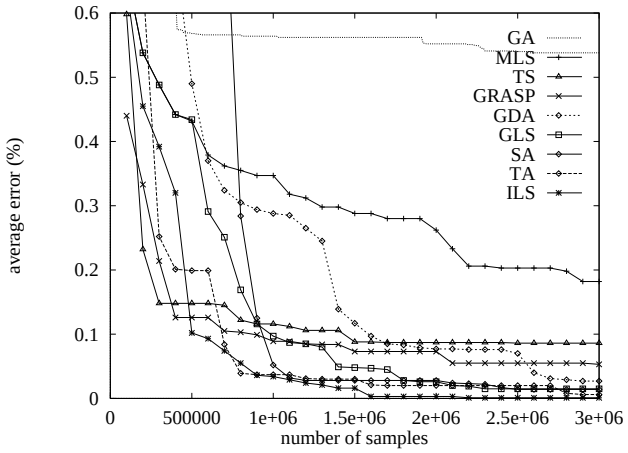

図 2: サンプル数に対する解の精度 (%) (近傍 N_{swap}, 突然変異 SWAP の場合).

図より, 以下のことが観測できる. (1) GA 法の性能は, 突然変異にあまり大きく左右されないが, 得られる解の精度はあまり高くない. (2) GRASP 法, ILS 法, GLS 法, SA 法, TA 法, および, GDA 法は, いずれも MLS 法よりも高い性能を示している. その中でも, 同じ近傍を用いた場合には, ILS 法, GLS 法, SA 法, TA 法, および, GDA 法が特に良好な結果を得ている. (3) GA 法以外の全ての方法の性能は, 近傍に大きく依存している.

6 まとめ

「できるだけ単純な枠組みに従い, アルゴリズム内部のオペレータには複雑なものを用いない」ことに留意しつつ, 様々なタイプのメタ戦略について計算実験を行った. その結果, メタ戦略を「手軽なツール」として利用する際の設計指針としては,

1. まず MLS 法を試み, その際近傍は十分工夫する,
2. より高い精度が要求される場合には, ILS 法を試みる,
3. ILS 法であまり効果が得られなかった場合には, GLS 法か SA 法を試みる,

のが良いと思われる. なお, 指針 3 には, 定評のある枠組みの一つとして SA 法を挙げたが, 本研究の実験結果からは, その変形である TA 法や GDA 法でも十分と思われる.

今後, タイプの異なる問題に対しても同様の実験を行う予定である.

なお, 本研究は一部文部省科学研究費の援助を得たものである.

参考文献

[1] E.H.L. Aarts and J.H.M. Korst, *Simulated Annealing and Boltzmann Machines*, (Wiley, 1989).

[2] E.H.L. Aarts, P.J.M. Van Laarhoven, J.K. Lenstra and N.L.J. Ulder, "A Computational Study of Local Search Algorithms for Job Shop Scheduling," *ORSA J. Computing*, 6 (1994) 118–125.

[3] I. Althöfer and K.-U. Koschnick, "On the Convergence of "Threshold Accepting"," *Applied Mathematics and Optimization*, 24 (1991) 183–195.

[4] K.R. Baker and G.D. Scudder, "Sequencing with Earliness and Tardiness Penalties: A Review," *Operations Research*, 38 (1990) 22–36.

[5] K.D. Boese, A.B. Kahng and S. Muddu, "A New Adaptive Multi-Start Technique for Combinatorial Global Optimizations," *Operations Research Letters*, 16 (1994) 101–113.

[6] R.M. Brady, "Optimization Strategies Gleaned from Biological Evolution," *Nature*, 317 (1985) 804–806.

[7] I. Charon and O. Hudry, "The Noising Method: a New Method for Combinatorial Optimization," *Operations Research Letters*, 14 (1993) 133–137.

[8] H.A.J. Crauwels, C.N. Potts and L.N. Van Wassenhove, "Local Search Heuristics for Single-Machine Scheduling with Batching to Minimize the Number of Late Jobs," *European J. Operational Research*, 90 (1996) 200–213.

[9] L. Davis, "Applying Adaptive Algorithms to Epistatic Domains," in: *Proc. 9th IJCAI*, (Morgan Kaufmann, 1985) p. 162–164.

[10] L. Davis, *Handbook of Genetic Algorithms*, (Van Nostrand Reinhold, 1991).

[11] G. Dueck, "New Optimization Heuristics: the Great Deluge Algorithm and the Record-to-Record Travel," Heidelberg Scientific Center Research Report TR89.06.011, IBM, Germany (1989).

[12] G. Dueck and T. Scheuer, "Threshold Accepting: A General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing," *J. Computational Physics*, 90 (1990) 161–175.

[13] T.A. Feo and J.L. González-Velarde, "The Intermodal Trailer Assignment Problem," *Transportation Science*, 29 (1995) 330–341.

[14] T.A. Feo and M.G.C. Resende, "A Probabilistic Heuristic for a Computationally Difficult Set Covering Problem," *Operations Research Letters*, 8 (1989) 67–71.

[15] T.A. Feo and M.G.C. Resende, "Greedy Randomized Adaptive Search Procedures," *J. Global Optimization*, 6 (1995) 109–133.

[16] T.A. Feo, M.G.C. Resende and S.H. Smith, "A Greedy Randomized Adaptive Search Procedure for Maximum Independent Set," *Operations Research*, 42 (1994) 860–878.

[17] T.A. Feo, K. Venkatraman and J.F. Bard, "A GRASP for a Difficult Single Machine Scheduling Problem," *Computers and Operations Research*, 18 (1991) 635–643.

[18] B. Freisleben and P. Merz, "A Genetic Local Search Algorithm for Solving Symmetric and Asymmetric Traveling Salesman Problems," in: *Proc. IEEE International Conference on Evolutionary Computation*, (1996) p. 616–621.

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

[20] F. Glover, "Tabu Search — Part I," *ORSA J. Computing*, 1 (1989) 190–206; Part II, ditto, 2 (1990) 4–32.

[21] F. Glover, "Tabu Search and Adaptive Memory Programming — Advances, Applications and Challenges," to appear in: *Interfaces in Computer Science and Operations Research*, (Kluwer Academic Publishers, 1996).

[22] F. Glover and M. Laguna, "Tabu Search," a chapter in: *Modern Heuristic Techniques for Combinatorial Problems*, (1992).

[23] F. Glover, E. Taillard and D. de Werra, "A User's Guide to Tabu Search," *Annals of Operations Research*, 41 (1993) 3–28.

[24] D.E. Goldberg and R. Lingle, "Alleles, Loci, and the Traveling Salesman Problem," in: *Proc. 1st ICGA*, (Lawrence Erlbaum Associates, 1985) p. 154–159.

[25] D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, (Addison-Wesley, 1989).

[26] J. Hart and A. Shogan, "Semi-Greedy Heuristics: An Empirical Study," *Operations Research Letters*, 6 (1987) 107–114.

[27] J.H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, (The University of Michigan Press, 1975, and MIT Press, 1992).

[28] 茨木俊秀, "組合せ最適化の手法 — 巡回セールスマン問題の例から —," 電学論 C, 114 (1994) 411–419.

[29] 茨木俊秀, "組合せ最適化とスケジューリング問題: 新解法とその動向," 計測と制御, 34 (1995) 340–346.

[30] T. Ibaraki, "Combination with Other Optimization Methods," a chapter in: *Handbook of Evolutionary Computation*, (IOP Publishing Ltd and Oxford University Press, 1996) (to appear).

[31] T. Ibaraki and Y. Nakamura, "A Dynamic Programming Method for Single Machine Scheduling," *European J. Operational Research*, 76 (1994) 72–82.

[32] P. Jog, J.Y. Suh and D. Van Gucht, "The Effects of Population Size, Heuristic Crossover and Local Improvement on a Genetic Algorithm for the Traveling Salesman Problem," in: *Proc. 3rd ICGA*, (Morgan Kaufmann, 1989) p. 110–115.

[33] P. Jog, J.Y. Suh and D. Van Gucht, "Parallel Genetic Algorithms Applied to the Traveling Salesman Problem," *SIAM J. Optimization*, 1 (1991) 515–529.

[34] D.S. Johnson, "Local Optimization and the Traveling Salesman Problem," in: *Proc. 17th Colloquium on Automata, Languages and Programming*, (Springer-Verlag, 1990) p. 446–461.

[35] D.S. Johnson, C.R. Aragon, L.A. McGeoch and C. Schevon, "Optimization by Simulated Annealing: An Experimental Evaluation; Part I, Graph Partitioning," *Operations Research*, 37 (1989) 865–892; "Part II, Graph Coloring and Number Partitioning," ditto, 39 (1991) 378–406.

[36] B.W. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs," *Bell System Technical Journal*, 49 (1970) 291–307.

[37] S. Kirkpatrick, C.D. Gelatt, Jr. and M.P. Vecchi, "Optimization by Simulated Annealing," *Science*, 220 (1983) 671–680.

[38] 北野宏明 (編), 遺伝的アルゴリズム, (産業図書, 1993); 第 2 巻, 同上, (1995).

[39] A. Kolen and E. Pesch, "Genetic Local Search in Combinatorial Optimization," *Discrete Applied Mathematics*, 48 (1994) 273–284.

[40] 久保幹雄, "メタヒューリスティックス," a chapter in: 離散構造とアルゴリズム IV, (近代科学社, 1995) p. 171–230.

[41] M. Laguna, T.A. Feo and H.C. Elrod, "A Greedy Randomized Adaptive Search Procedure for the Two-Partition Problem," *Operations Research*, 42 (1994) 677–687.

[42] Y. Li, P.M. Pardalos and M.G.C. Resende, "A Greedy Randomized Adaptive Search Procedure for the Quadratic Assignment Problem," *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, 16 (1994) 237–261.

[43] C.K.Y. Lin, K.B. Haley and C. Sparks, "A Comparative Study of both Standard and Adaptive Versions of Threshold Accepting and Simulated Annealing Algorithms in Three Scheduling Problems," *European J. Operational Research*, 83 (1995) 330–346.

[44] S. Lin and B.W. Kernighan, "An Effective Heuristic Algorithm for the Traveling Salesman Problem," *Operations Research*, 21 (1973) 498–516.

[45] M. Lundy and A. Mees, "Convergence of an Annealing Algorithm", *Mathematical Programming*, 34 (1986) 111–124.

[46] O. Martin, S.W. Otto and E.W. Felten, "Large-Step Markov Chains for the TSP Incorporating

Local Search Heuristic," *Operations Research Letters*, 11 (1992) 219–224.

[47] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, (Springer-Verlag, 1992).

[48] H. Mühlenbein, "Parallel Genetic Algorithms, Population Genetics and Combinatorial Optimization," in: *Proc. 3rd ICGA*, (Morgan Kaufmann, 1989) p. 416–421.

[49] H. Mühlenbein, M. Gorges-Schleuter and O. Krämer, "Evolution Algorithms in Combinatorial Optimization," *Parallel Computing*, 7 (1988) 65–85.

[50] I.M. Oliver, D.J. Smith and J.R.C. Holland, "A Study of Permutation Crossover Operators on the Traveling Salesman Problem," in: *Proc. 2nd ICGA*, (Lawrence Erlbaum Associates, 1987) p. 224–230.

[51] I.H. Osman and J.H. Kelly, "Meta-Heuristics: An Overview," in: *Meta-Heuristics: Theory and Applications*, (Kluwer Academic Publishers, 1996) p. 1–21.

[52] C.H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, (Prentice-Hall, 1982).

[53] C.N. Potts and L.N. Van Wassenhove, "A Decomposition Algorithm for the Single Machine Total Tardiness Problem," *Operations Research Letters*, 1 (1982) 177–181.

[54] M.G.C. Resende and T.A. Feo, "A GRASP for Satisfiability," *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, (to appear).

[55] M. Sinclair, "Comparison of the Performance of Modern Heuristics for Combinatorial Optimization on Real Data," *Computers and Operations Research*, 20 (1993) 687–695.

[56] J.Y. Suh and D. Van Gucht, "Incorporating Heuristic Information into Genetic Search," in: *Proc. 2nd ICGA*, (Lawrence Erlbaum Associates, 1987) p. 100–107.

[57] 玉置久, 喜多一, 岩本貴司, 三宮信夫, "遺伝アルゴリズム — I — GA の基礎," システム制御情報学会誌, 39 (1995) 295–302; 玉置久, 喜多一, "II — GA による最適化計算 1," 同上, 39 (1995) 400–407; 玉置久, 喜多一, "III — GA による最適化計算 2," 同上, 39 (1995) 431–438; 岩本貴司, "IV — GA の理論," 同上, 39 (1995) 507–514; 玉置久, 喜多一, 岩本貴司, "V — GA の拡張," 同上, 40 (1996) 69–76; 玉置久, 喜多一, 岩本貴司, "VI — 進化型計算の動向," 同上, 40 (1996) 170–177.

[58] N.L.J. Ulder, E.H.L. Aarts, H.-J. Bandelt, P.J.M. Van Laarhouven and E. Pesch, "Genetic Local Search Algorithms for the Traveling Salesman Problem," in: *Proc. 1st PPSN*, (Springer-Verlag, 1990) p. 109–116.

[59] M. Yagiura and T. Ibaraki, "Genetic and Local Search Algorithms as Robust and Simple Optimization Tools," in: *Meta-Heuristics: Theory and Applications*, (Kluwer Academic Publishers, 1996) p. 63–82.

[60] M. Yagiura and T. Ibaraki, "Metaheuristics as Robust and Simple Optimization Tools," in: *Proc. IEEE International Conference on Evolutionary Computation*, (1996) p. 541–546.

[61] M. Yagiura and T. Ibaraki, "The Use of Dynamic Programming in Genetic Algorithms for Permutation Problems," *European J. Operational Research*, 92 (1996) 387–401.