日本ファジィ学会誌 Vol.12, No.4, pp.562-569(2000)

論文

多次元整数ナップサック問題に対する2重構造文字列遺伝的アルゴリズム†

坂和 正敏*1　加藤 浩介*1　柴野 俊弘*2　廣瀬 公彦*1

本論文では，多次元整数ナップサック問題に対して，近年，最適化，適応，学習のための方法論として注目されてきている遺伝的アルゴリズムによる近似解法を提案する．多次元整数ナップサック問題に対しては，坂和らにより環状2重構造文字列遺伝的アルゴリズム，あるいは，3重構造文字列遺伝的アルゴリズムによる解法が提案されてきているが，いまだ発展途上である．そこで，本論文では，精度の向上を目指して連続緩和問題の解の情報を取り入れた2重構造文字列表現を用いた遺伝的アルゴリズムを提案し，数値実験によりその有効性を示す．

キーワード 多次元整数ナップサック問題，遺伝的アルゴリズム，2重構造文字列，連続緩和問題

1. はじめに

整数計画問題は組合せ最適化問題であり，規模の増大にともなって，探索すべき解の個数が爆発的に増加する．このため，現時点では，一般に，大規模な整数計画問題を厳密に解くことは困難であり，効率的な近似解法が望まれている．

遺伝的アルゴリズムは，自然界のシステムにおける生物の進化のメカニズムを模擬するアルゴリズムとして，1970年代初頭にJ.H. Hollandによって提唱され，最適化の新しい方法論として近年活発に研究が行われている[2-4]．ところで，遺伝的アルゴリズムを最適化問題に適用する際にしばしば問題になるのが，制約の取り扱いである．一般には，実行可能解のみが探索対象となるように探索空間及び遺伝演算子を設計する方法，デコーディングの手続きで実行可能解に変換する方法，制約条件を満たさない度合をペナルティとして適合度を下げる方法などによって制約条件に対処することになる[8]．

坂和らは，多次元0-1ナップサック問題に対して，すべての個体を実行可能解に対応づけるための2重構造文字列表現と対応するデコーディングアルゴリズムを用いた遺伝的アルゴリズム[5]を提案し，有効性を示している．また，その拡張として，多次元整数ナップサック問題に対する2重構造文字列を用いた遺伝的アルゴリズム[6]を提案しているが，精度及び計算時間に関して改善が望まれていた．さらに，彼らは，多次元整数ナップサック問題に対する3重構造文字列を用いた遺伝的アルゴリズムによる近似解法[7]を提案し，精度に関して2重構造文字列を用いた遺伝的アルゴリズムに対する優位性を示したが，残念ながらこれも決定的な手法とはなり得ていない．

そこで，本研究では，2重構造文字列遺伝的アルゴリズムに焦点をあて，連続緩和問題の解の情報を用いることにより，整数ナップサック問題の広大な制約領域における探索の効率化を図り，解の精度の向上を目指す．

2. 多次元整数ナップサック問題

一般に，多次元整数ナップサック問題は次のように定式化される．

$$\left.\begin{array}{l}\text{minimize } \boldsymbol{cx}\\ \text{subject to } A\boldsymbol{x}\leq\boldsymbol{b}\\ \qquad x_j\in\{0,\ldots,\nu_j\},\ j=1,\ldots,n\end{array}\right\}\quad(1)$$

ここで，$\boldsymbol{c}=(c_1,\ldots,c_n)$は$n$次元行ベクトル，$\boldsymbol{x}=(x_1,\ldots,x_n)^T$は$n$次元整数決定変数列ベクトル，$\boldsymbol{b}=(b_1,\ldots,b_m)^T$は$m$次元列ベクトル，$A=(a_{ij})$は$m\times n$係数行列，$\nu_j, j=1,\ldots,n$は正の整数である．また，$\boldsymbol{c}$の各成分はすべて負であるとともに，$A,\boldsymbol{b}$の各成分はすべて非負であり，$X$は問題(1)の実行可能領域を表すものとする．

この多次元整数ナップサック問題は一つの組合せ最適化問題であり，実用的な規模の問題に適用可能な厳密解法及び近似解法は確立されているとはいえず，発展途上である．このような組合せ最適化問題の近似解法の一つとして，遺伝的アルゴリズムが近年注目されてきており，組合せ最適化問題への適用に関する研究

† Genetic Algorithms with Double Strings for Multidimensional Integer Knapsack Problems

Masatoshi SAKAWA, Kosuke KATO, Toshihiro SHIBANO and Kimihiko HIROSE

*1 広島大学工学部第二類(電気系)
Department of Industrial and Systems Engineering, Faculty of Engineering, Hiroshima University

*2 新菱冷熱工業㈱都市設備事業部
Urban Facilities Division, Shinryo Corporation

が活発に行われている[2-4].

遺伝的アルゴリズムを最適化問題に適用する際にしばしば問題になるのが制約の取り扱いである. 坂和らは,すでに多次元0-1ナップサック問題に対して提案していた, すべての個体を実行可能解に対応づけるための2重構造文字列表現と対応するデコーディングアルゴリズムを用いた遺伝的アルゴリズム[5]の拡張として,多次元整数ナップサック問題に対する2重構造文字列を用いた遺伝的アルゴリズム[6]を提案している.そこでは,図1のような2重構造文字列表現と2重構造文字列により表現された個体をすべて実行可能解と対応づけることができるデコーディングアルゴリズムが用いられている.

2重構造文字列のデコーディングアルゴリズム

手順1： $j:=1$, $\mathrm{sum}_i:=0$, $i=1,\ldots,m$ とおく.

手順2： $a_{is(j)}\neq 0$のところで

$$x_{s(j)}:=\min\left(\min_{i=1,\ldots,m}\left\lfloor\frac{b_i-\mathrm{sum}_i}{a_{is(j)}}\right\rfloor, g_{s(j)}\right)$$

とする.

手順3： $\mathrm{sum}_i:=\mathrm{sum}_i+a_{is(j)}x_{s(j)}$, $i=1,\ldots,m$ として, $j:=j+1$とする.

手順4： $j>n$ ならば終了. そうでなければ手順2へ戻る.

このような2重構造文字列による個体表現と対応するデコーディングアルゴリズムを用いた遺伝的アルゴリズムにおいては, 上段の添字の順列及び下段の変数の値の候補は初期においてランダムに決定され, 世代が進むにつれてより望ましい並びが得られるようになる. しかしながら, 整数計画問題のように0-1計画問題に比べて探索空間がきわめて大きい問題においては, 最適な並びだけでなく最適な決定変数の値を同時に得ることは非常に困難であると考えられ, 精度及び計算時間に関して改善が望まれていた. そこで, 彼らは, 多次元整数ナップサック問題に対する3重構造文字列を用いた遺伝的アルゴリズムによる近似解法[7]を提案し,精度に関して2重構造文字列を用いた遺伝的アルゴリズムに対する優位性を示したが,3重構造文字列遺伝的アルゴリズムでは, デコーディングアルゴリズムや交叉オペレータが整数ナップサック問題に特化してかなり複雑化している, あるいは, 個体表現自体にパラメータが含まれるというように決定すべきパラメータが2重構造文字列遺伝的アルゴリズムに比べて多い, というような理由から拡張性及び発展性に関しては不利であると考えられ, 残念ながらこれも決定的な手法とはなり得ていない.

Indices	$s(1)$	$s(2)$	$\cdots$	$s(j)$	$\cdots$	$s(n)$
Values	$g_{s(1)}$	$g_{s(2)}$	$\cdots$	$g_{s(j)}$	$\cdots$	$g_{s(n)}$

図1 2重構造文字列. $s(j)\in\{1,2,\ldots,n\}$は変数の添字, $g_{s(j)}\in\{0,1,\ldots,\nu_{s(j)}\}$は変数$x_{s(j)}$の値の候補を表す.

このような状況の下で, 本研究においては, 2重構造文字列遺伝的アルゴリズムに焦点をあて, 連続緩和問題を解いて得られた解の情報を利用することにより, 精度の改善を試みる.

3. 整数ナップサック問題の最適解と連続緩和問題の最適解の関係

ここでは, 一つの例として, 50変数10制約の単一目的の整数ナップサック問題をランダムに20個生成し, それらの最適解 $\boldsymbol{x}^*=(x_1^*,\ldots,x_{50}^*)^T$と対応する連続緩和問題(線形計画問題)の最適解 $\hat{\boldsymbol{x}}=(\hat{x}_1,\ldots,\hat{x}_{50})^T$を求め, これら解の関係について調べた.

問題の具体的な生成法は, $\boldsymbol{c}$ の要素 c_j, $j=1,\ldots,50$ は$[-999,0]$の一様乱数, A の要素 a_{ij}, $i=1,\ldots,10$, $j=1,\ldots,50$は$[0,999]$の一様乱数により決定され, $\boldsymbol{b}$ の要素は

$$b_i:=\gamma\cdot\sum_{j=1}^{n}a_{ij},\quad i=1,\ldots,10 \tag{2}$$

により決定され, γ は$[10,20]$の一様乱数により決定された. また, 変数 x_j, $j=1,\ldots,50$の上限 ν_jはすべて20とした.

20個の整数ナップサック問題に対する最適解 $\boldsymbol{x}^*$と対応する連続緩和問題の最適解 $\hat{\boldsymbol{x}}$ の各要素について, $x_j^*=\hat{x}_j$となった回数を $x_j^*=0$の場合とそうでない場合と分けて調べた結果を表1に示す.

さらに, 図2に $\hat{x}_j$と x_j^*の差 $\hat{x}_j-x_j^*$の度数分布を示す.

この実験から, 整数ナップサック問題の最適解は連続緩和問題の最適解に比較的近い値をとるということが示唆され, 特に, 連続緩和問題の最適解において0の値をとる変数は整数ナップサック問題の最適解においても0になりやすいという性質があることがわかる.

このような考察に基づいて, 次節では, 連続緩和問題の最適解の情報を取り入れた2重構造文字列遺伝的アルゴリズムを提案する.

表1 整数ナップサック問題の最適解 x_j^* と連続緩和問題の最適解 $\hat{x}_j$ の関係

	$x_j^* = \hat{x}_j$	$x_j^* \neq \hat{x}_j$
$x_j^* = 0$	328	2
$x_j^* \neq 0$	585	85

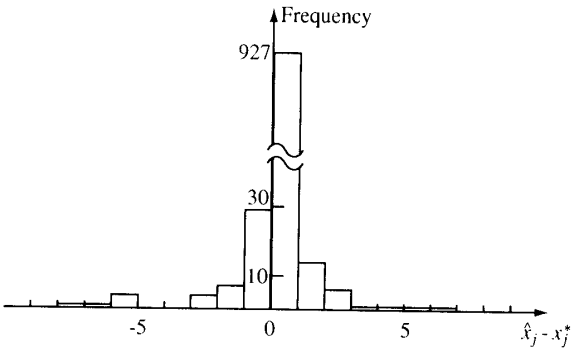


図2 整数ナップサック問題の最適解 x_j^* と連続緩和問題の最適解 $\hat{x}_j$ の差 $\hat{x}_j - x_j^*$ の度数分布

4. 連続緩和問題の最適解の情報を利用した遺伝的アルゴリズム

4.1 連続緩和問題の最適解の情報を利用したデコーディングアルゴリズム

坂和らによって提案されている図1のような2重構造文字列に対するデコーディングアルゴリズムでは，文字列の左側からデコードされるため，遺伝子 $(s(j), g_{s(j)})^T$ は文字列の左端に近ければ近いほどデコードにおける優先順位が高く，$g_{s(j)}$ に近い値にデコードされやすくなる一方で，右端に近い方では $g_{s(j)}$ よりも小さい値にデコードされやすい．そこで，本論文では，整数ナップサック問題の解と連続緩和問題の解の関係を考慮して，連続緩和問題を解いて得られた解において0よりも大きい値をとる決定変数について優先的にデコーディングを行い，デコーディングアルゴリズムにより対応づけられる解を最適解に近づけることによって，遺伝的アルゴリズムにより得られる近似最適解の精度の向上を図る．

提案デコーディングアルゴリズム

手順1： $j := 1$, $\mathrm{sum}_i := 0$, $i = 1, \ldots, m$ とする．

手順2： もし，$x_{s(j)} > 0$ ならば，手順3へ進む．そうでなければ，つまり $x_{s(j)} = 0$ ならば，$j := j+1$ として手順5へ進む．

手順3： $a_{is(j)} \neq 0$ のところで

$$x_{s(j)} := \min\left(\min_{i=1,\ldots,m} \left\lfloor \frac{b_i - \mathrm{sum}_i}{a_{is(j)}} \right\rfloor, g_{s(j)} \right)$$

とする．

手順4： $\mathrm{sum}_i := \mathrm{sum}_i + a_{is(j)} x_{s(j)}$, $i = 1, \ldots, m$ とし，$j := j+1$ とする．

手順5： もし $j > n$ ならば，手順6へ進み，そうでなければ，手順2へ戻る．

手順6： $j := 1$ とする．

手順7： もし，$x_{s(j)} = 0$ ならば，手順8へ進む．そうでなければ，つまり $x_{s(j)} > 0$ ならば，$j := j+1$ として手順10へ進む．

手順8： $a_{is(j)} \neq 0$ のところで

$$x_{s(j)} := \min\left(\min_{i=1,\ldots,m} \left\lfloor \frac{b_i - \mathrm{sum}_i}{a_{is(j)}} \right\rfloor, g_{s(j)} \right)$$

とする．

手順9： $\mathrm{sum}_i := \mathrm{sum}_i + a_{is(j)} x_{s(j)}$, $i = 1, \ldots, m$ とし，$j := j+1$ とする．

手順10： もし $j > n$ ならば，終了．そうでなければ，手順7へ戻る．

ここで，整数ナップサック問題の連続緩和問題の最適解 $\hat{\boldsymbol{x}}$ はあらかじめ計算されているものとする．

図3に，2重構造文字列で表現された個体のデコーディングの例を示す．2節で述べた従来のデコーディングアルゴリズムによれば，図中のように左から順にデコードされるが，提案デコーディングアルゴリズムを用いた場合は，左から順に，$x_{s(j)} > 0$ を満たすところをデコードしたのち，$x_{s(j)} = 0$ であるところをデコードする．

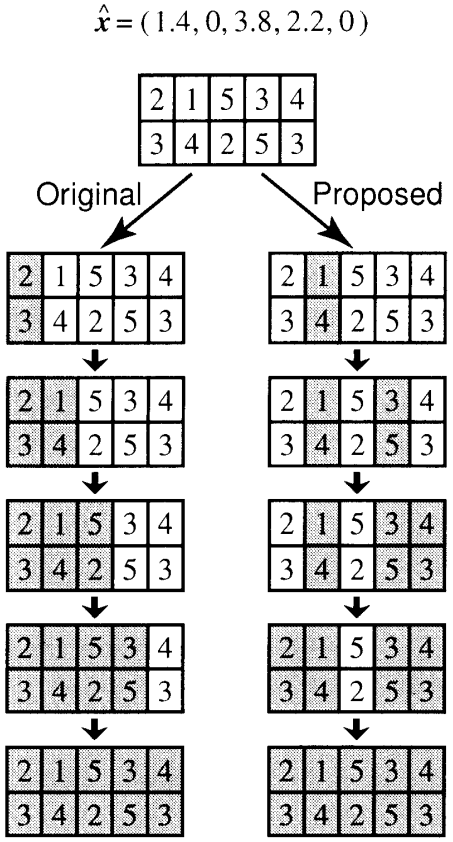


図3 デコーディングの例

4.2 連続緩和問題の最適解の情報を利用した初期個体群の生成

従来の2重構造文字列を用いた遺伝的アルゴリズム[6]においては，下段の $g_{s(j)}$ の値は，初期的に $\{0,1,\ldots,v_j\}$ の乱数により決定されていた．しかしながら，もし初期個体群を最適解の近くに発生させることが可能ならば，より最適解に到達しやすく，望ましいと考えられる．そこで，本論文では，整数ナップサック問題の最適解は連続緩和問題を解いて得られた解に比較的近い値になるという性質を利用した初期個体群の発生方法を提案する．

具体的には，連続緩和問題の最適解を $\hat{x}_j$, $j=1,\ldots,n$ としたときに，各初期個体の下段 $g_{s(j)}$, $j=1,\ldots,n$ の値を，平均 $\hat{x}_{s(j)}$，標準偏差 σ の正規分布に従う乱数により決定する．

4.3 連続緩和問題の最適解の情報を利用した突然変異

従来の2重構造文字列遺伝的アルゴリズム[6]では，下段の $g_{s(j)}$ に対して，$\{0,1,\ldots,v_{s(j)}\}$ の値をランダムに発生させて，その値を突然変異後の値としていた．しかし，突然変異においてもより最適解の近くに変化させた方がより効率的であると考えられる．そこで，本論文では，初期個体群の発生の場合と同様に連続緩和問題の最適解の情報を利用した突然変異オペレータを提案する．

具体的には，連続緩和問題の最適解を $\hat{x}_j$, $j=1,\ldots,n$ としたときに，突然変異オペレータが適用される個体の下段の要素 $g_{s(j)}$ の値を，平均 $\hat{x}_{s(j)}$，標準偏差 ρ の正規分布に従う乱数により決定する．

突然変異の例を図4に示す．

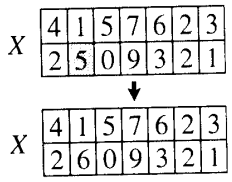


図4 突然変異の例

4.4 適合度

本論文では各個体 $\mathbf{S}$ の適合度を

$$f(\mathbf{S})=\frac{\boldsymbol{cx}}{\sum_{j=1}^{n} c_j \cdot v_j} \tag{3}$$

と定義する．ここで，$\sum_{j=1}^{n} c_j \cdot v_j$ は，制約がない場合の目的関数の最小値を表しているので，適合度関数は目的関数値を目的関数の最小値で正規化したものとなり，$0 \le f(\mathbf{S}) \le 1$ なる関係が成立していることがわかる．

4.5 スケーリング

各個体に対する適合度関数が決定されたとき，この適合度関数による評価値そのものをそのまま選択時の確率に反映させるよりはむしろ，なんらかの関数を導入して適合度関数の差異を適切に拡大あるいは縮小させる方がより効果的な選択が行われ，アルゴリズムがより効果的に働くような状況も考えられる．このような考えに基づいて，提案されたのがスケーリングである．本研究では，最も一般的なスケーリング法である線形スケーリングを採用する．

線形スケーリングのアルゴリズム

手順1： 個体群の平均適合度 f_{mean}，最大適合度 $f_{\max}$，最小適合度 $f_{\min}$ を求める．

手順2： もし

$$f_{\min} > \frac{c_{\mathrm{mult}} \cdot f_{\mathrm{mean}} - f_{\max}}{c_{\mathrm{mult}} - 1.0} \tag{4}$$

ならば手順3へ，そうでなければ手順4へ進む．

手順3：

$$a := \frac{(c_{\mathrm{mult}} - 1.0) \cdot f_{\mathrm{mean}}}{f_{\max} - f_{\mathrm{mean}}}, \tag{5}$$

$$b := \frac{f_{\mathrm{mean}} \cdot (f_{\max} - c_{\mathrm{mult}} \cdot f_{\mathrm{mean}})}{f_{\max} - f_{\mathrm{mean}}} \tag{6}$$

として手順5へ進む．

手順4： $a := \frac{f_{\mathrm{mean}}}{f_{\mathrm{mean}} - f_{\min}}$, $b := -\frac{f_{\min} \cdot f_{\mathrm{mean}}}{f_{\mathrm{mean}} - f_{\min}}$ として手順5へ進む．

手順5： $i=1,2,\ldots,N$ まで $f' := a \cdot f_i + b$ なる線形スケーリングを行う．

ここで，c_{mult} は最良の個体が次世代に残す個対数の期待値を表し，通常は $1.2 \le c_{\mathrm{mult}} \le 2.0$ を満たすように設定される[4]が，この数値により遺伝的アルゴリズムの性能が大きく左右されるので，重要な遺伝的パラメータの一つである．

4.6 再生

再生にはさまざまな方法が考えられるが，坂和らは[5]において，6種類のオペレータ(切断選択，エリート切断選択，期待値選択，エリート期待値選択，ルーレット選択，エリートルーレット選択)についての性能の比較検討を行っており，エリート期待値選択モデルが比較的有効な手法であることを示している．したがって，本論文においても，次世代に残る個体の期待値により再生を行う期待値選択に，現在の個体の中で最大の適合度を持つ個体は無条件で次世代に残すエリート保存選択を加えた，エリート期待値選択を採用する．

期待値選択

純粋な確率的選択では，個体数が十分に多くない時

には，乱数の揺らぎによって適合度を正確に反映しない選択がなされる可能性がある．このような問題点に対処するために提案された期待値選択(expected-value selection)は，個体群の中の各個体の適合度とその総計を計算し，適合度の総計に対する各個体の割合で個体数を調整するという基本的な考えに基づいている．

具体的には，N 個体からなる個体群に対して，i 番目の個体 $\mathbf{S}_i$ の次世代の期待個体数は

$$N_i=\left(\frac{f(\mathbf{S}_i)}{\sum_{i=1}^{N} f(\mathbf{S}_i)}\right)\times N \tag{7}$$

のように計算される．このとき，N_i の整数部分($\lfloor N_i \rfloor$)が各個体の確定的な次世代の個体数であるものとする．一方，N_i の小数部分($N_i-\lfloor N_i \rfloor$)は，次世代に1個体が生存する確率とみなし，$N-\sum_{i=1}^{N}\lfloor N_i \rfloor$ の個体をこの確率によって決定する．

エリート保存選択

確率にしたがって個体を選択して交叉や突然変異を行う場合には，非常に良い個体が現れてもすぐに消滅してしまうことがある．このことは，確率的な操作をする以上やむを得ないことであり，局所解に陥ることを避けることにもつながるが，現実に少ない回数で良い解を得たい場合には好ましくない．そこで，個体群の中で最も適合度の高い個体は無条件でそのまま次世代に残すというエリート保存選択(elitist preserving selection)が提案されており，「$\mathbf{S}^*_{(t)}$ を，世代 t までに現れた最良の個体とする．もし，$X(t+1)$ を通常の方法で生成したときに，$X(t+1)$ の中に，$\mathbf{S}^*_{(t)}$ が存在しなければ，$\mathbf{S}^*_{(t)}$ を $X(t+1)$ の $N+1$ 番目の個体として加える．」のように定義されている．

エリート選択を採用すれば，その時点での最良の個体は，交叉や突然変異により破壊されることはないという利点がある．しかし，エリート個体の遺伝子が個体群の中に急速に広がる可能性が高いので局所解に陥る危険も含んでいる．したがって，エリート選択は，他の選択手法と組み合わせて使用されることが多い．このようなエリート保存選択を用いた場合には，個体群の中の適合度の最大値は単調非減少になる．

4.7 交叉

交叉は，個体群の中から任意の二つの個体(親)をランダムに選び，更に，ランダムに選ばれた一点あるいは多点の交叉点で文字列の一部を組み替えることにより，新たな二つの個体(子)を生成する操作である．本論文では，次に述べるような2重構造文字列に対する部分一致交叉(partially matched crossover：PMX)[4]を採用する．

部分一致交叉(PMX)

手順0： 2つの親個体を X, Y とし，そのコピーをそれぞれ X', Y' とする．

手順1： X', Y の i 番目の添字の要素を $s_{X'}(i)$, $s_Y(i)$，対応する変数の値を $g_{s_{X'}(i)}$, $g_{s_Y(i)}$ とする．交叉点をランダムに2点選んで，交換する部分文字列を定め，手順2へ進む．

手順2： 交換する部分文字列の先端と末端を h と k, $1\le h<k\le n$ とし，$j:=h$ とおく．

手順3： $s_Y(j)=s_{X'}(j')$ となる j' を求め，X' の j 番目の要素 $(s_{X'}(j), g_{s_{X'}(j)})^T$ と j' 番目の要素 $(s_{X'}(j'), g_{s_{X'}(j')})^T$ と交換して，$j:=j+1$ とする．

手順4： $j>k$ ならば終了，そうでなければ手順3に戻る．

手順5： 得られた X' の h から k の部分の文字列を Y の h から k の部分の文字列と交換し，この X' を X の子個体とする．Y' に対しても同様の操作を行い，Y の子個体とする．

部分一致交叉(PMX)の例を図5に示す．

4.8 逆位

ある長さの部分文字列の順序を反転させるという，次のような逆位オペレータも使用する．

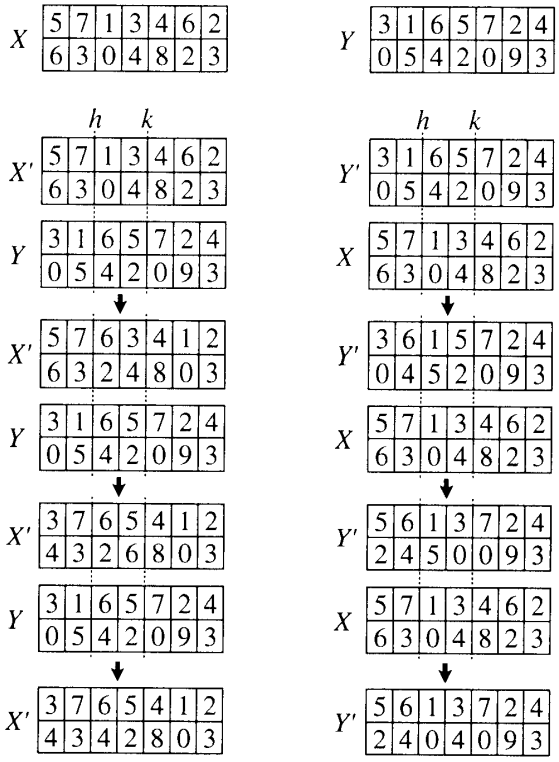


図5 部分一致交叉(PMX)の例

逆位のアルゴリズム

手順1： 2点 $h,k,1\leq h<k\leq n$ をランダムに選び，2重構造文字列の h 番目から k 番目の部分文字列を抜き出す．

手順2： h 番目から k 番目の部分文字列の要素を逆方向に並べ換える．

手順3： 逆方向に並べ換えられた部分文字列をもとの2重構造文字列に戻して終了する．

逆位の例を図6に示す．

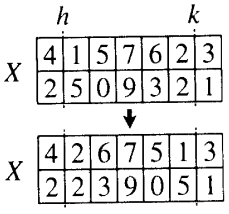


図6 逆位の例

4.9 提案遺伝的アルゴリズムの概要

手順0(パラメータ設定) 反復回数(世代)を表す変数 t の初期化($t:=0$)，及び，個体群サイズ N，交叉率 p_c，突然変異率 p_m，逆位率 p_i，最大探索世代 $I_{\max}$，スケーリング定数 c_{mult} の設定を行う．

手順1(初期個体群生成) 初期個体群として N 個の個体を4.2節で述べた方法にしたがって生成する．

手順2(評価) 各個体 $\mathbf{S}$ を4.1節で述べたデコーディングアルゴリズムにより実行可能解 $\boldsymbol{x}$ にデコードした後，各個体の適合度 $f(\mathbf{S})$ を4.4節にしたがって求める．

手順3(終了判定) $t>I_{\max}$ が成り立っているならば終了する．そうでなければ，$t:=t+1$ として手順4に進む．

手順4(再生) 4.6節で述べたエリート期待値選択により再生を行う．

手順5(交叉) 交叉率 p_c にしたがって，4.7節で述べた部分一致交叉(PMX)を行う．

手順6(突然変異) 突然変異率 p_m にしたがって，4.3節で述べた突然変異を行う．

手順7(逆位) 逆位率 p_i にしたがって，4.8節で述べた逆位を行う．手順2に戻る．

5. 数値実験

本提案手法の有効性を検証するために，整数ナップサック問題に対して，提案遺伝的アルゴリズム(提案GA)と従来の2重構造文字列遺伝的アルゴリズム(従来GA)[6]及びM. Berkelaarらの分枝限定法に基づくソフトウェアLp_solve[1]を適用する．

数値実験に用いた遺伝的アルゴリズムのパラメータはすべて共通として，それぞれ，個体群サイズ100，最大探索世代500，交叉率0.8，突然変異率0.05，逆位率0.01，スケーリング定数2.0と設定し，初期個体群の生成においては平均 $\bar{x}_j$，標準偏差 $\sigma=1$ の正規分布に従う乱数，突然変異においては平均 $\bar{x}_j$，標準偏差 $\rho=3$ の正規分布に従う正規分布を用いて値を決定している．

ここで，σ と ρ の値についての設定に関する適当な指針や目安は残念ながら得られていないので，いくつかの σ と ρ の設定の中で比較的良い結果を与えた $\sigma=1$，$\rho=3$ を採用した．また，目的関数の係数 c_j，$j=1,\ldots,n$ と制約式左辺の係数 a_{ij}，$i=1,\ldots,m$，$j=1,\ldots,n$ はそれぞれ $\{-100,-99,\ldots,-1\}$，$\{0,1,\ldots,999\}$ の一様乱数により決定され，制約式の右辺定数 b_i，$i=1,\ldots,m$ は左辺の係数の和 $\sum_{j=1}^{n} a_{ij}$ に5をかけて得られたものを用いているとともに，変数の値の上限 v_j はすべて30と設定されている．

50変数20制約の問題に対する適用結果を表2，80変数25制約の問題に対する適用結果を表3，100変数30制約の問題に対する適用結果を表4に示す．遺伝的アルゴリズムによる結果としては，10回の試行における最良目的関数値(最良値)，平均目的関数値(平均値)，最悪目的関数値(最悪値)，平均計算時間(秒)，平均収束世代を示している一方で，Lp_solveによる結果としては，10800秒までに得られた暫定解の目的関数値を記している．また，参考のため，連続緩和問題の最適目的関数値も各表の最下行に示している．ここで，収束世代は各試行における最良解が求まった世代を表す．また，実験を行う計算機環境は，CPU：Intel Celeron Processor 333MHz，OS：Microsoft Windows2000，C_Compiler： Microsoft Visual C++ 6.0である．

表2～4から，提案GAは，従来GAよりやや長い計算時間を要しているが，収束世代が従来GAに比べて早いので探索の早期の打ち切りが可能であることを考慮すれば，両手法の計算時間は同程度であると考えられる．一方，解の精度に関しては，最良値，最悪値，平均値のすべてにおいて従来GAから大きな改善がみられている．これらのことから，提案GAが従来GAより近似解法として優れていることが示唆される．また，分枝限定法に基づいたLp_solveとの比較においても，提案GAは1～2%以下の計算時間で，より優れた近似解を得ているとともに，連続緩和問題の最適目的関数値との相対誤差が1%以下であることから高精度の近似解が得られていることもわかる．したがって，解の精度の改善を目指して本論文で提案している連続

表2 50変数20制約の問題に対する適用結果

	最良値	平均値	最悪値	時間（秒）	平均収束世代
提案 GA	−21149	−21131.5	−21115	6.38×10^1	262.1
従来 GA	−20592	−20231.8	−19645	6.21×10^1	408.0
Lp_solve	−21045（暫定値）			1.08×10^4	—
連続緩和問題	−21205.069			—	—

表3 80変数25制約の問題に対する適用結果

	最良値	平均値	最悪値	時間（秒）	平均収束世代
提案 GA	−34530	−34444.3	−34384	1.26×10^2	121.9
従来 GA	−33470	−32399.2	−31788	1.17×10^2	382.9
Lp_solve	−33539（暫定値）			1.08×10^4	—
連続緩和問題	−34623.214			—	—

表4 100変数30制約の問題に対する適用結果

	最良値	平均値	最悪値	時間（秒）	平均収束世代
提案 GA	−46287	−46168.8	−46123	1.89×10^2	136.9
従来 GA	−43762	−42477.2	−41217	1.73×10^2	402.2
Lp_solve	−44881（暫定値）			1.08×10^4	—
連続緩和問題	−46465.448			—	—

緩和問題の最適解の情報を利用した遺伝的アルゴリズムは近似解法として有用であると考えられる．

6. おわりに

本論文では，整数ナップサック問題に対して，広い解空間を効率的に探索するために，連続緩和問題の最適解の情報をデコーディング，初期個体群の発生及び突然変異に取り入れた2重構造文字列表現を用いた遺伝的アルゴリズムを提案した．いくつかの数値実験により，提案する連続緩和問題の解の情報を利用した2重構造文字列遺伝的アルゴリズムに基づく解法，従来の2重構造文字列遺伝的アルゴリズムに基づく解法[6]及び分枝限定法に基づく解法[1]の比較を行い，提案手法の有効性を示した．今後は，意思決定者の判断のあいまい性をとりいれたファジィ多目的整数ナップサック問題に対する対話型ファジィ満足化手法等への拡張を試みる予定である．

参 考 文 献

[1]M. Berkelaar, LP_SOLVE, ftp：//ftp.es.ele.tue.nl/pub/lp_solve.

[2]M. Gen and R. Cheng, Genetic Algorithms and Engineering Design, John Wiley & Sons, 1996.

[3]Z. Michalewicz, Genetic Algorithm+Data Structures=Evolution Programs, Springer-Verlag, Berlin, 1992, Third revised and extended edition, 1996.

[4]坂和正敏，田中雅博，遺伝的アルゴリズム，朝倉書店，1995.

[5]M. Sakawa, K. Kato, H. Sunada and T. Shibano, Fuzzy programming for multiobjective 0-1 programming problems through revised genetic algorithms, European Journal of Operational Research, Vol.97, pp.149-158, 1997.

[6]坂和正敏，柴野俊弘，加藤浩介，多目的整数計画問題に対する遺伝的アルゴリズムによる対話型ファジィ満足化手法，日本ファジィ学会誌，Vol.10, No.1, pp.108-116, 1998.

[7]坂和正敏，加藤浩介，柴野俊弘，宮原伸二，多目的整数ナップザック問題に対する3重構造文字列遺伝的アルゴリズムによる近似解法，電子情報通信学会論文誌，Vol.J82-A, No.5, pp.691-697, 1999.

[8]三宮信夫，喜多一，玉置久，岩本貴司，遺伝アルゴリズムと最適化，朝倉書店，1998.

(1999年12月22日 受 付)
(2000年 5月18日 再受付)

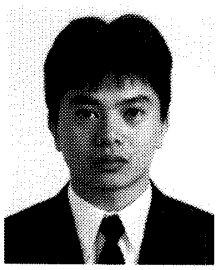
[問い合わせ先]
〒739-8527
東広島市鏡山1-4-1
広島大学工学部第二類(電気系)
計数管理工学講座
坂和 正敏
TEL：0824(24)7694, (22)7159
FAX：0824(24)7694, (22)7195
E-Mail：sakawa@msl.sys.hiroshima-u.ac.jp

著者紹介

さかわ まさとし
坂和 正敏 [正会員]

広島大学 工学部第二類(電気系)

昭和45年 京大・工・数理卒，昭和50年同大大学院博士課程了．昭和50年 神戸大・工・システム助手，昭和56年 同助教授，昭和62年 岩手大・工・教授を経て，平成2年 広島大・工・第二類(電気系)計数管理工学講座教授となり現在に至る．大規模システム，多目的システム，ファジィシステムにおける意思決定手法とその応用に関する研究に従事．工博．著書「線形システムの最適化」，「非線形システムの最適化」，「ファジィ理論の基礎と応用」，「経営数理システムの基礎」，「数理計画法の基礎」，「離散システムの最適化」(森北出版)，「ソフト最適化」，「遺伝的アルゴリズム」(朝倉書店)，Fuzzy Sets and Interactive Multiobjective Optimization (Plenum Press)等．



かとう こうすけ
加藤 浩介 [正会員]

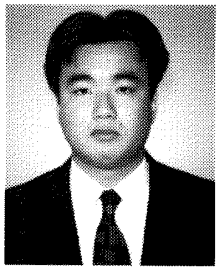
広島大学 工学部第二類(電気系)

平成3年 阪大・基礎工・生物卒．平成5年 同大大学院博士課程前期了．平成6年同後期退学．平成6年 広島大・工・第二類(電気系)計数管理工学講座助手となり現在に至る．博士(工学)．ソフトコンピューティング手法によるシステム最適化およびシステム同定に関する研究に従事．

しばの としひろ
柴野 俊弘 [正会員]

新菱冷熱工業㈱ 都市設備事業部

昭和54年 室蘭工大・工・電気卒．昭和54年 新菱冷熱工業(株)に入社し現在に至る．ファジィ制御，遺伝的アルゴリズムおよびファジィ計画法などの研究に従事．博士(工学)．日本ファジィ学会，計測自動制御学会，電気学会，空気調和・衛生工学会各会員．



ひろせ きみひこ
廣瀬 公彦

広島大学 工学部第二類(電気系)

平成9年 広島大・工・第二類(電気系)卒．平成11年 同大大学院博士課程前期了．平成11年 富士通株式会社に入社し現在に至る．広島大在学中は遺伝的アルゴリズムを用いたファジィ多目的整数計画法に関する研究に従事．

Genetic Algorithms with Double Strings for Multidimensional Integer Knapsack Problems

by

Masatoshi SAKAWA, Kosuke KATO, Toshihiro SHIBANO and Kimihiko HIROSE

Abstract :

In this paper, for multidimensional integer knapsack problems we propose an approximate solution method through genetic algorithms which have recently attracted considerable attention in a number of fields as a methodology for optimization, adaptation and learning. For the multidimensional integer knapsack problems, M. Sakawa et al. proposed approximate solution methods through genetic algorithms using ringed double string representation or triple string representation, but more development and improvement are desired to them. Thus, in this paper, we propose a genetic algorithm using double string representation based on a solution for the corresponding continous relaxation problem to improve the accuracy or precision of solutions. Furthermore, the efficiency and effectiveness of the proposed method will be shown through various numerical experiments.

Keywords : Multidimensional Integer Knapsack Problems, Genetic Algorithm, Double String Representation, Continuous Relaxation Problem

Contact Address : **Masatoshi SAKAWA**
Faculty of Engineering, Hiroshima University
4-1, *Kagamiyama* 1-*chome*, *Higashi-Hiroshima*, 739-8527 *Japan*
TEL : 0824(24)7694, (22)7159
FAX : 0824(24)7694, (22)7195
E-mail : sakawa@msl.sys.hiroshima-u.ac.jp