探索アルゴリズムその1

Group F

提出日:平成17年月日()

Member
045709E：上原直久
045713C：大城和也
045734B：知念栄作
045739G：友寄雄一朗

1　課題内容

2　Level1 コンピュータと人間の違いを述べよ。

2.1　コンピュータが人より得意なもの

2.1.1　マニュアル通りに動く

コンピュータが得意なものとして、事務的な処理が挙げられる。データの入力などの速度は人とは比較にならない。また、バグが起こらない限りその作業が正確であることも挙げられる。また、その作業を特定の場所に保存できる。

つまり、事務的処理は会社のマニュアル通りにこなす必要があるので、マニュアル通りに動くコンピュータに人は勝ることはできない。コンピュータがマニュアル通りに動く理由は、コンピュータがプログラム通りに動く性質があるからである。

2.1.2　正確な記憶

コンピュータに勝る記憶量を持った人は存在しないと言える。なぜなら、人は記憶を取り出せなくなってしまうためである。コンピュータは容量があれば、どんな些細なことでも忘れること無い。その理由として、大量の情報から必要な情報を取り出す検索機能が優れているためだと考えられる。

2.2　コンピュータが人より苦手なもの

2.2.1　独創性、創造性

コンピュータは、様々なプログラムを動かすことができるので、何でもできると思われがちだが、実際はプログラムにあることしか処理することはできない。つまり、プログラム通りにしかコンピュータは動くことしかできない。よって、コンピュータで想定外のものは作り出せないので、独創性、創造性が無いと考えられる。

2.2.2　曖昧

コンピュータは曖昧なことは苦手だと考えられる。それは、コンピュータ内部では 0 と 1 で情報を表しているので、曖昧なことも 0 と 1 で表現しようとするからである。つまり、数で厳密に区別できないものが苦手なので、曖昧なことが苦手だと考えられる。

3　3.3 のナップザック問題

3.1　計算困難な問題

今回の Level3.2，3.3 の問題はそれぞれ一般的にナップザック問題，巡回セールスマン問題と呼ばれており，どちらも最適な値を求めるにはそれぞれの組み合わせを全探索するしかなく，その探索には全ての組み合わせを調べなければならなくなる．この時の計算量は n が増えるごとに爆発的に上昇し，現実的には不可能に近くなる．ナップザック問題は NP 完全，巡回セールスマン問題は NP 困難に部類されている．

3.2　ナップザック問題

ナップザック問題とはいくつかの荷物と一定容量の荷物を入れれる袋がある．この時，容量を超えないで価値の和が最大になる組み合わせを決める問題のことである．先でも述べたようにナップザック問題は荷物が n この場合それぞれを入れる入れないを考えるので，解の候補は 2^n となり，この問題の完璧な最適化を行うには全探索をする必要がある．

3.2.1　F グループ案

私たち F グループはナップザック問題においてある程度最適なものを選べ，現実的な速度で探索する方法としてヒューリスティック関数を用いる方法を考えた．その方法を以下に示す。

- まず，荷物１つ１つの評価値を求めるためのヒューリスティック関数を考える．
- ヒューリスティック関数からそれぞれの評価値を求め，評価を行う．
- 上位、いくつかを選出し、その組み合わせでナップザック問題を考える．
- その中で一番，価値が高くできたものをこの問題の解と考える。.

具体的に考えるならば、まず荷物 i(i=1,$\cdots$,7) とし，その重量を $g_i > 0$，価値を $e_i > 0$ として許容重量を M とする．このとき，それぞれが下のようになるとする．

$$\begin{aligned} g &= (2,\ 5,\ 3,\ 1,\ 3,\ 6,\ 4) \\ e &= (50,\ 60,\ 18,\ 10,\ 9,\ 24,\ 20) \\ M &= 10 \end{aligned}$$

この時，評価値を kg 単位の価値とするようにするつまりヒューリスティック関数 h は

$$h = \frac{e}{g}$$

となる。それより $\frac{e}{g}$={25,12,6,10,3,4,5} が導きだされ，評価値が高かった上位 5 つを選出し，並び替えを行いなが比較し，最適化を行う．またこのとき，荷物 i を袋に入れるかそうでないかを $x_i \in \{0, 1\}$ で表し，それぞれの値を並べて 7 ビットの 2 進数記号として扱う．下位となったものは常に 0 にするようにする．つまり，このときなら 2 桁目と 3 桁目は常に 0 となるようにする．ここで最適化を行う際には 10 以下という重量の制限を加えておく．

それらを踏まえて 5 つの並び替えをし最適化をすると 1110000 というビットが出る．これは重さ 10 で価値 128 という値となる．

この方法を扱うことでその計算量は O(2^n) から O(n) となる．(ただし、上位の選出の数が log(n) 個以内のとき) しかし，初めに書いたとおりこの方法では確実に最適化を行えることはない，特に値が大きくなると切られるビットも多くなり，そのビットに最適な値に含まれる場合には大きな誤差がでることが考えられる．そこで，この考えとともに遺伝的アルゴリズムを考えてみた．

3.2.2 遺伝的アルゴリズム

GA(Genetic Algorithm) とは生物の進化の過程を模倣したアルゴリズムのことでにある．GA では，探索点を複数個用い、各探索点が，遺伝子をもつ仮想的な生物であるとみなす．各個体に対して，それぞれ環境との適応度を計算する．低い適応度をもつ個体を淘汰して消滅させ，高い適応度をもつ個体を増殖させ，親の形質を継承した遺伝子をもつ子孫の個体を生成する世代交代シミュレーションを実行する．この時，遺伝子の交差および突然変異と呼ばれる操作を行う．そして，最終的に，非常に高い適応度の個体ができる，これを最適値とみるのである．この一連の GA の流れを図 3.2.2 に乗せる

では，先ほどのアルゴリズムにこの考えを付け足す．まず，ヒューリスティックを用いた評価値からそれぞれの発生率を導きだす (評価値が高いものほど選ばれる確率が高くなる) ここで，この発生率のもと多数のサンプルを作り出す．これを遺伝子型と考える．その遺伝子の中で適応度が低いものを淘汰するようにする．今回の場合だと

$$\text{適応度} \quad = \quad \sum_{i=1}^{n} e_i x_i - \alpha \; max\{0, \sum_{i=1}^{n} g_i x_i - M\}$$

となる．これは価値の合計を求めるものだが，その際の合計の重さが制限の M より大きい場合には値が引かれる．このとき，合計の重さを超えたものを淘汰するために $\alpha > 0$ は大きくするのがよいと思われる．

適応度が高くて選ばれた遺伝子型は，複数点交叉 (遺伝子ビットの相互の交代) による遺伝子型の継承 (図 1) と，低確率で起こる突然変異 (図 2) によるビットの反転を行ってその子供を作る．そしてまた，適応度が調べられ，交叉，突然変異する．このプロセスを繰り返すことで最適化を行う．

これにより，先の案での不安要素であるビットの抜け落ちなども考慮でき，また単に遺伝的アルゴリズムを用いるより早く最適化が可能だと考えられる．

図 1: 2 点交叉

図 2: 突然変異

図 3: GA のフローチャート

参考文献

[1] “http://www.ohanashi.jp/it/2.htm” コンンピュータの得意技

[2] “http://www.algolab.co.jp/ lum/pcnyumon/nyu041.htm”
迷宮からの脱出

[3] ジェネティックアルゴリズム, 著者：安居院 猛　長尾 智晴

[4] 遺伝アルゴリズムと最適化, 著者：三宮 信夫, 喜多 一, 玉置 久, 岩本 貴司

[5] ”http://www.as.media.yamanashi.ac.jp/h̃attori/demo/ga.html”
Genetic Algorithms