1　ナップサック問題

乱数を使った計算手法の例として、組み合わせ最適化問題を解く方法を示します。組み合わせの最適化問題の典型例として、ナップサック問題を取り上げます。
ナップサック問題は、制限重量のあるナップサックに、価値と制限重量が決まっている複数の荷物を詰め込む問題です。
ナップサック問題では、ナップサックの制限重量を超えることなく、なるべく荷物の価値の合計値が大きくなるように荷物をつめなければなりません。ある荷物のセットとナップサックに対して、価値が最大となる荷物の組み合わせを最適解あるいは厳密解とよびます。
ナップサック問題の最適解を解析的に求める方法は知られていないので、基本的には、最適解を求めるためにはすべての組み合わせを調べなければなりません。荷物がひとつ増えるごとに組み合わせの数が 2 倍になるので、力ずくの総当たり法では、荷物の数が数十程度までしか求めることができません。そこで探索の方法を工夫する手段として、一般に動的計画法や分枝限定法などが用いられます。ここでは、最適解を求める代わりに、乱数を使って比較的優良な解を求めることを考えます。

C による数値計算とシュミレーション　小高　知宏　オーム社　より

```
0001: /*
0002:   rkp.c
0003:   ナップサック問題をランダム探索で解くプログラムです。
0004:   使い方　./rkp（荷物の個数）（制限重量）（試行回数）
0005: */
0006:
0007: #include <stdio.h>
0008: #include <stdlib.h>
0009:
0010: #define BUFFSIZE 256
0011: #define MAXP 256
0012: #define SEED RAND_MAX-1
0013: #define R 10
0014:
0015: void initparcel(int parcel[] [2]);
0016: void prints(int p[], int n);
0017: void solve(int parcel[][2], int p[], int weightlimit,
0018:            long nlimit, int n);
0019: void rsetp(int p[], int n);
0020: double frand(void);
0021: int    calcval(int parcel[][2], int p[], int n);
0022: int    calcw(int parcel[][2], int p[], int n);
0023: int main(int argc, char **argv)
```

```
0024: {
0025:   int  parcel[MAXP][2] = {0};
0026:   int  p[MAXP] = {0};
0027:   int  weightlimit;
0028:   int  n;
0029:   long nlimit;
0030:   int  r;
0031:
0032:   if(argc != 4){
0033:     fprintf(stderr, "使い方 ./ rkp (荷物の個数) (制限
重量) (試行回数)\n");
0034:     exit(1);
0035:   }
0036:
0037:   n = atoi(argv[1]);
0038:   if((n <= 0) || (n > MAXP)){
0039:     fprintf(stderr, "荷物の個数が不正です(%d) \n", n);
0040:     exit(1);
0041:   }
0042:   printf("荷物の個数:%d\n", n);
0043:
0044:   if((weightlimit = atoi(argv[2])) <= 0){
0045:     fprintf(stderr, "制限重量が不正です(%d) \n", weightlimit);
0046:     exit(1);
0047:   }
0048:   printf("制限重量:%d\n", weightlimit);
0049:
0050:   if((nlimit = atoi(argv[3])) <= 0){
0051:     fprintf(stderr, "試行回数が不正です(%d) \n", nlimit);
0052:     exit(1);
0053:   }
0054:   printf("試行回数:%d\n", nlimit);
0055:
0056:   initparcel(parcel);
0057:   srand(SEED);
0058:   for(r = 0; r < R; ++r){
0059:     solve(parcel, p, weightlimit, nlimit, n);
0060:   }
0061:
0062:   return 0;
0063: }
0064:
0065: void solve(int parcel[][2], int p[], int weightlimit,
0066:            long nlimit, int n)
0067: {
0068:   int  maxvalue = 0, mweight;
0069:   int  value, weight;
0070:   long i;
0071:   int  bestp[MAXP] = {0};
```

```
0072:   int  j;
0073:
0074:   for(i = 0; i < nlimit; ++i){
0075:     rsetp(p, n);
0076:     if((weight = calcw(parcel, p, n)) <= weightlimit){
0077:       value = calcval(parcel, p, n);
0078:     }else{
0079:       value = 0;
0080:     }
0081:     if(value > maxvalue){
0082:       maxvalue = value;
0083:       mweight  = weight;
0084:       for(j = 0; j < n; ++j){
0085:         bestp[j] = p[j];
0086:       }
0087:     }
0088:   }
0089:   printf("%d  %d ", maxvalue, mweight);
0090:   prints(bestp, n);
0091: }
0092:
0093: int calcw(int parcel[][2], int p[], int n)
0094: {
0095:   int i;
0096:   int w = 0;
0097:
0098:   for(i = 0; i < n; ++i){
0099:     w += parcel[i][0] * p[i];
0100:   }
0101:
0102:   return w;
0103: }
0104:
0105: int calcval(int parcel[][2], int p[], int n)
0106: {
0107:   int i;
0108:   int v = 0;
0109:
0110:   for(i = 0; i < n; ++i){
0111:     v += parcel[i][1] * p[i];
0112:   }
0113:
0114:   return v;
0115: }
0116:
0117: void rsetp(int p[], int n)
0118: {
0119:   int i;
0120:
```

```
0121:   for(i = 0; i < n; ++i){
0122:     while((p[i] = frand() * 2) >= 2);
0123:   }
0124: }
0125:
0126: double frand(void)
0127: {
0128:   return (double)rand() / RAND_MAX;
0129: }
0130:
0131: void prints(int p[], int n)
0132: {
0133:   int i;
0134:   for(i = 0; i < n; ++i){
0135:     printf("%1d", p[i]);
0136:   }
0137:   printf("\n");
0138: }
0139:
0140: void initparcel(int parcel [MAXP][2])
0141: {
0142:   int i = 0;
0143:   char linebuf[BUFFSIZE];
0144:
0145:   while((i < MAXP) && (fgets(linebuf, BUFFSIZE, stdin) != NULL)){
0146:     sscanf(linebuf, "%d %d", &parcel[i][0], &parcel[i][1]);
0147:     ++i;
0148:   }
0149: }
```

2 重量と価値の表　kpq10.txt

```
87 96
66 55
70 21
25 58
33 41
24 81
89 8
63 99
23 59
54 62
```

3 「rkp」の操作方法

「rkp」の操作方法

./rkp（荷物の個数）（制限重量）（試行回数）<（荷物の重量と価値の表）

上記の「./」は、Linux でのプログラムを実行する時の例です。
ウィンドウズでは、rkp（荷物の個数）（制限重量）（試行回数）<（荷物の重量と価値の表）「Enter」と入力してください。

4 実行結果

荷物の個数:10
制限重量:250
試行回数:200
400 222 0001110111
331 235 1101110000
342 197 0000110111
397 246 1001110011
375 232 1001110100
393 234 0101110110
400 222 0001110111
353 231 1001100110
397 246 1001110011
397 246 1001110011

たまたま、厳密解 (価値の合計 400) が見つかっています。