BAB III

METODE PENELITIAN

3.1. Waktu dan Tempat Penelitian

Penelitian dilakukan dilingkungan Jurusan Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lampung. Waktu penelitian dilaksanakan pada semester 8 tahun ajaran 2014/2015.

3.2. Lingkungan Pengembangan

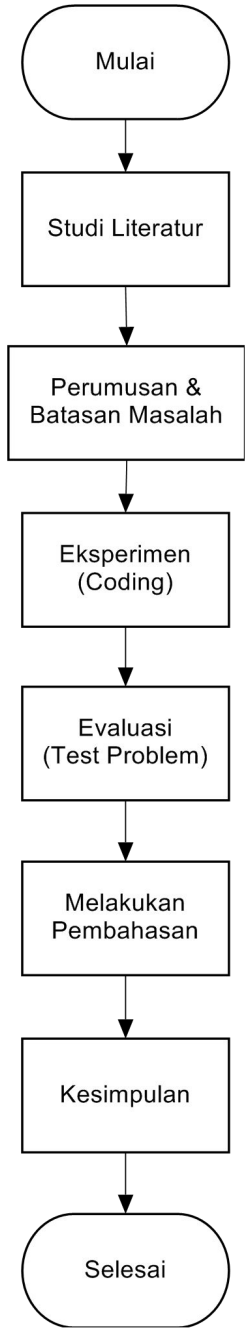
Lingkungan pengembangan yang akan digunakan pada penelitian ini adalah sebagai berikut:

- Perangkat lunak: OS Windows 7 Ultimate 32 bit, Matlab R2009a.
- Perangkat keras : Laptop Acer Aspire V5-471, Processor Intel(R) Core(TM) i3-2365M CPU @1.40GHz, RAM 2 GB.

3.3. Tahapan Penelitian

Kualitas keilmuan terlihat dari hasil penelitian yang diperoleh berdasarkan pada proses-proses penelitian yang telah direncanakan dengan baik. Proses penelitian

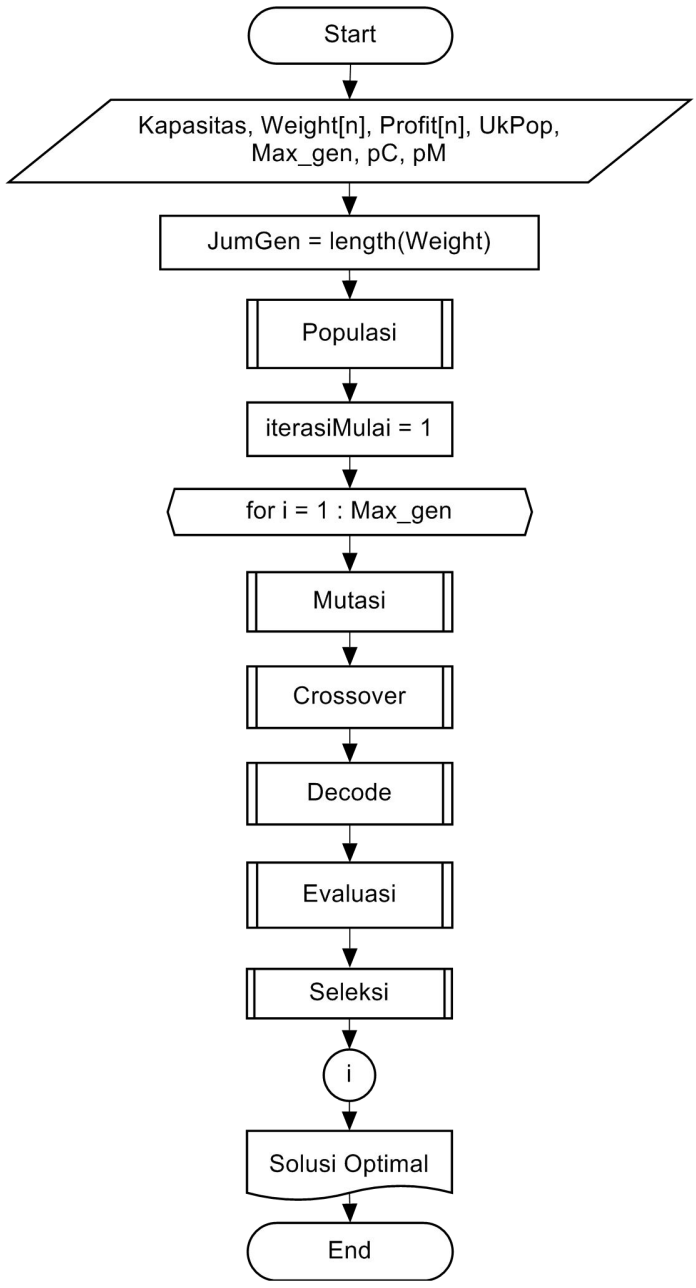
digunakan agar hasil penelitian mencapai optimasi pada berbagai keputusan riset. Gambar 3.1. menjelaskan bagaimana proses penelitian ini dilaksanakan.



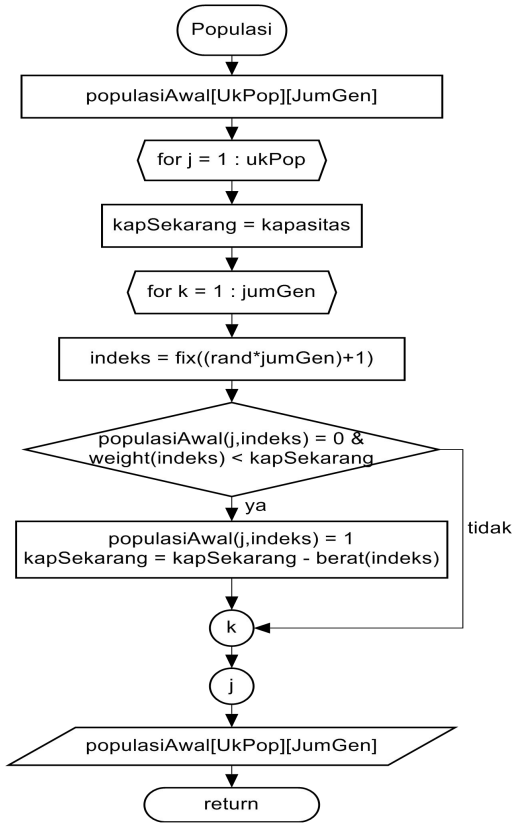
Gambar 3.1. Alur Penelitian.

3.3.1. Perancangan Algoritma

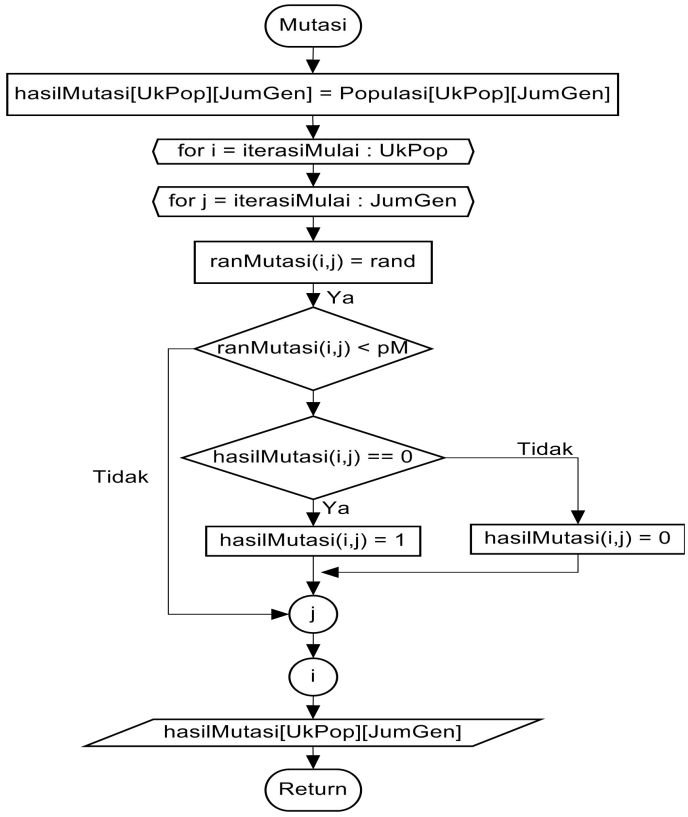
Pada tahap ini, dibuat suatu model GA untuk menyelesaikan *Knapsack problem*. Pada implementasinya hanya terdapat tiga tahap, yaitu *input* data, proses perhitungan *Knapsack problem* dengan GA dan *output* solusi. Rancangan GA ditunjukkan pada Gambar 3.2.



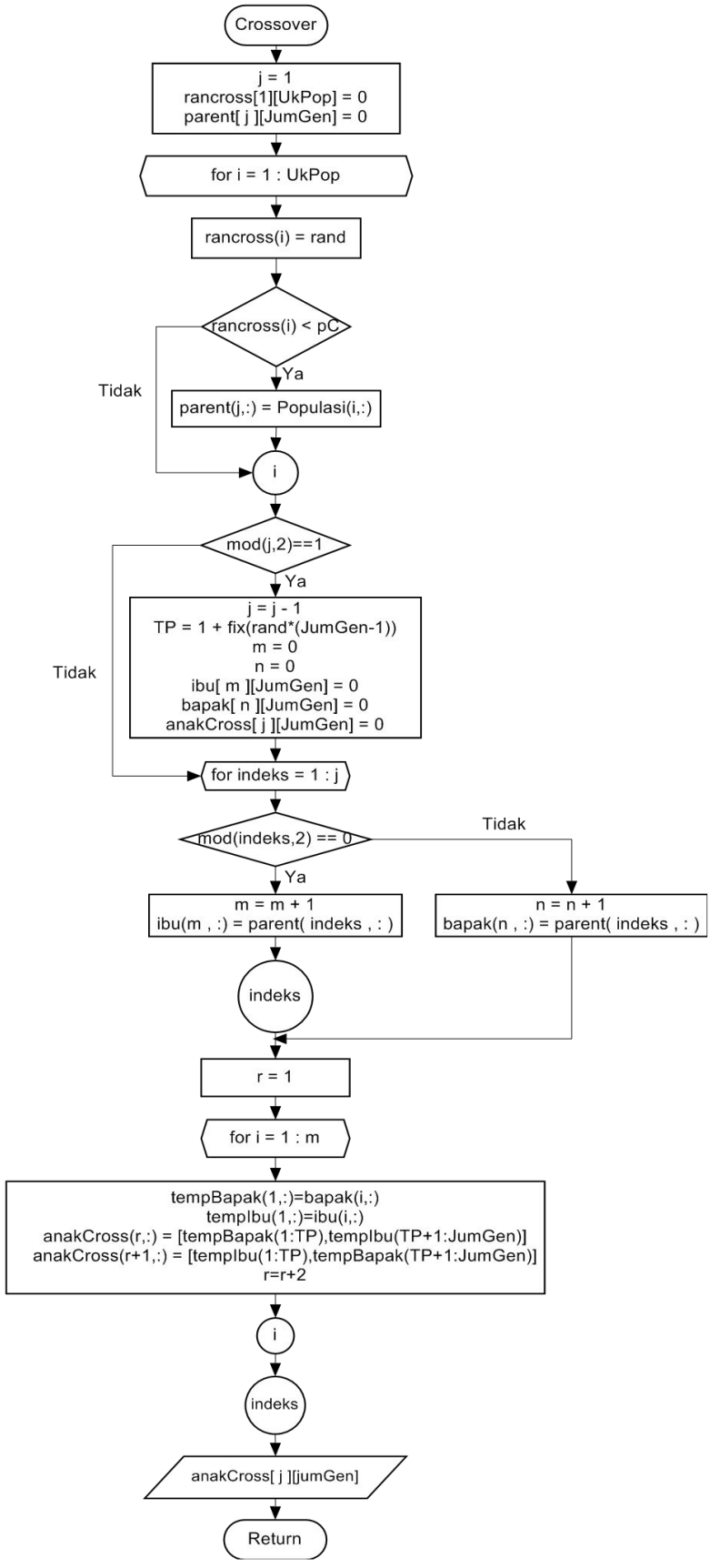
Gambar 3.2. Proses Optimasi *Knapsack Problem* dengan GA.



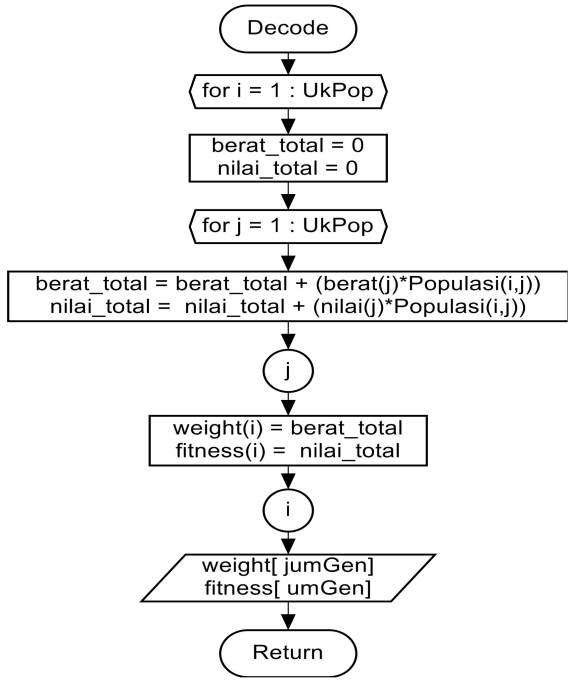
Gambar 3.3. Alur Pembangkitan Kromosom



Gambar 3.4. Alur Mutasi



Gambar 3.5. Alur *Crossover.*



Gambar 3.6. Alur *Decode*.

Jika diketahui total *weight* dari *item* terpilih melebihi kapasitas maka nilai penalti akan dibangkitkan menggunakan persamaan-persamaan berikut.

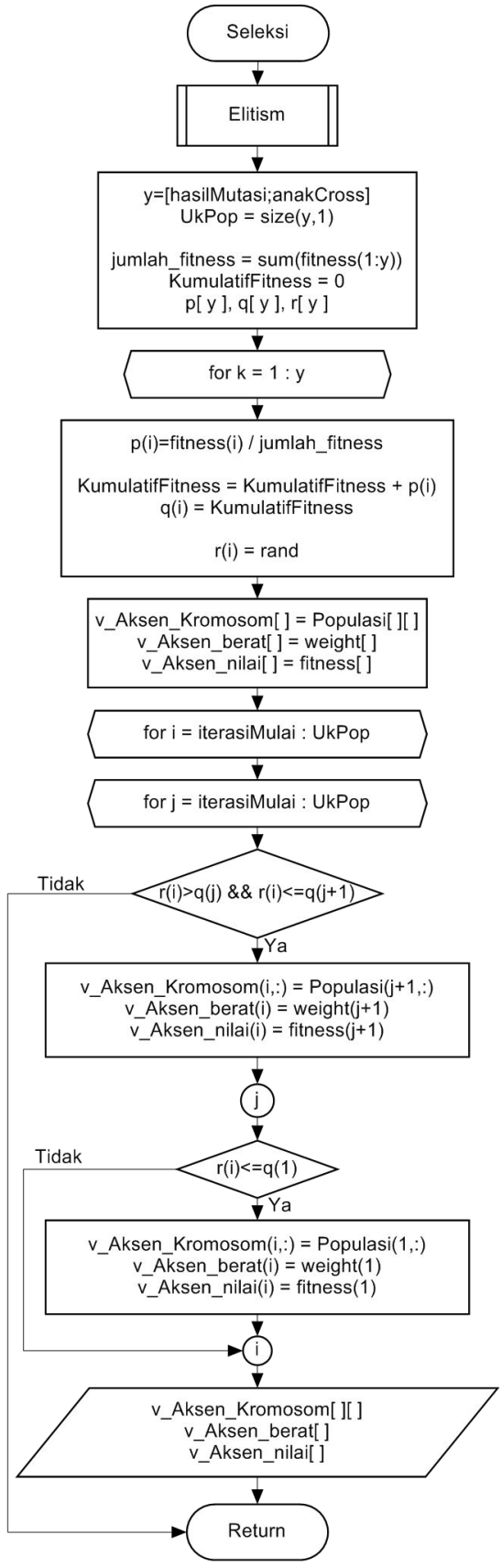
$$a = \sum_{i=1}^{n} w_i \qquad (3.1)$$

$$b = \sum_{i=1}^{n} w_i x_i \qquad (3.2)$$

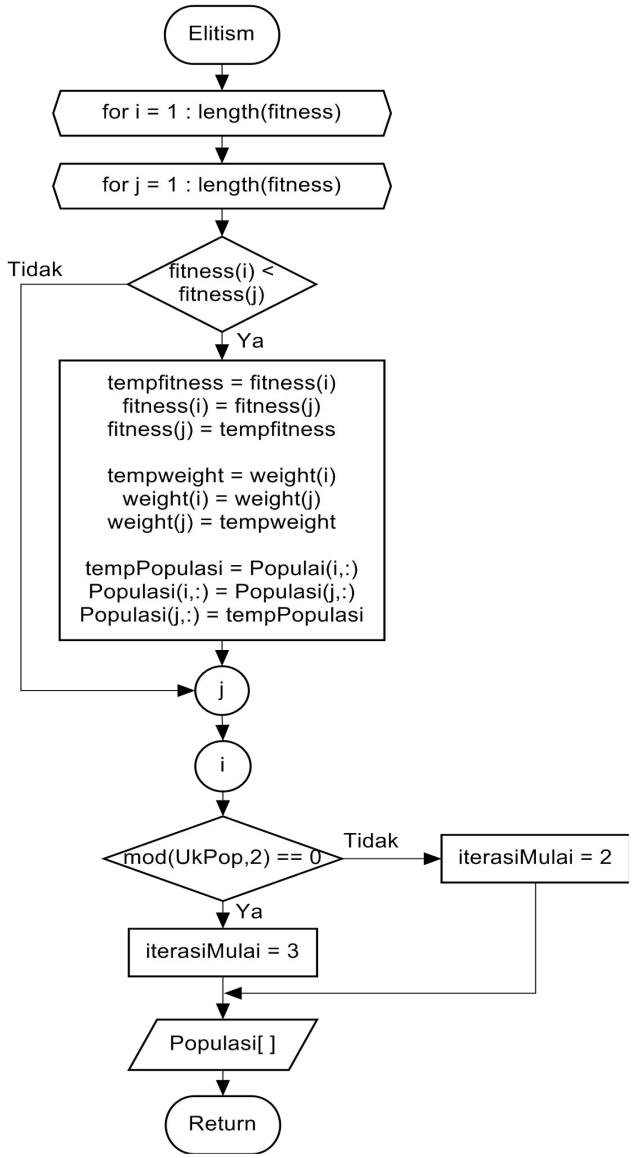
$$diff = \min\{c, |a - c|\} \qquad (3.3)$$

$$dist = |b - c| \qquad (3.4)$$

```
Prosedur strategi penalti :
if b < c then
    begin
        penalti = 1 - dist/diff;
        if penalti <= 0 then
            penalti = 0.00001;
            fit_val(i)= fit_val(i) * penalti;
      end
    end
end
```



Gambar 3.7. Alur Seleksi.



Gambar 3.8. Alur *Elitism*.

3.3.2. Implementasi

Proses GA diawali dengan membangkitkan populasi awal yang direpresentasikan oleh *bit string* (0 dan 1) secara langsung pada ruang solusi *feasible* atau secara acak. Pada *Knapsack problem* ruang solusi *feasible* merupakan kromosom-kromosom yang dibangkitkan langsung pada daerah yang dianggap layak, yaitu kromosom-kromosom yang memenuhi *constraint* atau dengan kata lain kromosom-kromosom yang memiliki total berat *item* tidak melebihi

kapasitas *Knapsack*. Tiap populasi berisi 1000 kromosom. Satu kromosom berisikan gen-gen yang berjumlah sesuai dengan banyaknya *item*.

1	2	3	4	5	6	7	8	9	10
0	1	1	0	1	0	0	0	0	1

Gambar 3.9. Representasi Kromosom pada *Knapsack Problem*.

Gambar 3.9. merupakan representasi kromosom pada *Knapsack problem* dengan *item* berjumlah 10 *item*. Pada gen yang berisi bit 1, yaitu *item* ke-2, 3, 5 dan 10 merupakan *item* terpilih sebagai *item* yang dimasukkan ke dalam *Knapsack,* sedangkan gen yang berisi bit 0 mewakili beberapa *item* yang tidak terpilih untuk dimasukkan kedalam *Knapsack*. Berikut ini langkah-langkah proses GA untuk *Knapsack problem* :

a. Merepresentasi kromosom pada *Knapsack problem* menggunakan *binary encoding*. *Binary encoding* merupakan pengkodean yang sering digunakan. Setiap kromosom berisikan *string* yang hanya terdiri dari bit 0 dan 1. Meskipun begitu tipe ini tidak begitu cocok pada beberapa permasalahan.
b. Membangkitkan populasi awal. Pada penelitian ini, dilakukan pembangkitan kromosom dengan dua tipe. Tipe pertama, *directed* GA yaitu pembangkitan kromosom langsung pada ruang solusi *feasible*. Tipe kedua pembangkitan kromosom secara *random*. Kromosom dibangkitkan sebanyak 1000 kromosom, satu kromosom terdiri dari gen-gen yang merepresentasikan solusi atau sejumlah *item*.
c. Memanipulasi kromosom dengan cara mutasi dan *crossover*. Kromosom dimanipulasi dengan *crossover* dan mutasi berdasarkan pada *probability crossover* = 0.80 dan *probability mutation* = 0.20.

d. Melakukan *decode* dengan cara menghitung *fitness value* dan total *weight* dari *item* terpilih. *Fitness value* dihitung berdasarkan fungsi tujuan seperti pada persamaan 2.1. Fungsi tujuan *Knapsack problem* ialah memaksimalkan *profit*.
e. Mengevaluasi masing-masing kromosom dengan strategi penalti. Jika *profit* maksimum dan total *weight* dari *item* terpilih ≤ kapasitas, pada kondisi ini kromosom dinyatakan memenuhi *constraint* atau solusi yang layak (*feasible*), maka *fitness value* dikalikan dengan nilai penalti $p(x)=1$. Sedangkan pada kromosom yang tidak memenuhi *constraint* atau solusi yang tidak layak (*infeasible*), *fitness value* dikalikan dengan nilai penalti $0 < p(x) \leq 1$. Nilai penalti berada pada interval $0 < p(x) \leq 1$, agar ketika *fitness value* dikalikan dengan nilai penalti, *fitness value* akan menjadi lebih kecil.
f. Melakukan tahap seleksi menggunakan *Elitism* dan *Roulette Wheel*. Proses ini dilakukan agar kromosom yang layak dapat lolos ke generasi selanjutnya.
g. Melakukan iterasi sebanyak maksimum generasi.

3.3.3. Eksperimen (Pengujian)

Tahap pengujian menggunakan *benchmark test problem.* Setiap *test problem* memiliki beberapa tipe data dengan jumlah data yang berbeda-beda. Parameter GA seperti probabilitas *crossover* dan mutasi, maksimum generasi ditentukan secara berurutan masing-masing 0,80 dan 0,20 serta 1000 generasi.

Data yang digunakan berupa data *weight, profit* dan kapasitas *Knapsack* yang berasal dari *website Kpacking* yang dipublikasikan oleh Claudio Tanci dari Universitas Perugia. (*n* = jumlah *item*).

Tabel 3.1. Deskripsi Data*.

No.	Test Problem	*n*	Optimum
1	p01	10	309
2	p02	5	51
3	p03	6	150
4	p04	7	107
5	p05	8	900
6	dataset20_1000	20	4129
7	small	40	1149
8	medium	100	1173
9	p08	24	13549094

*https://code.google.com/p/kpacking/source/browse/trunk.