PENCARIAN *MAXIMUM CLIQUE* MENGGUNAKAN ALGORITMA *IMPROVED ANT COLONY OPTIMIZATION* DAN *PARTICLE SWARM OPTIMIZATION* PADA JARINGAN SOSIAL

TUGAS AKHIR

Oleh :

ALEXANDER YAP
NIM. 121110341

JEFFRY OWEN
NIM. 121110227

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER
MIKROSKIL
MEDAN

2016

FINDING *MAXIMUM CLIQUE* USING *IMPROVED ANT COLONY OPTIMIZATION* AND *PARTICLE SWARM OPTIMIZATION ALGORITHM* IN SOCIAL NETWORKS

FINAL RESEARCH

By :

ALEXANDER YAP
ID. 121110341

JEFFRY OWEN
ID. 121110227

STUDY PROGRAM OF INFORMATICS ENGINEERING
SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER
MIKROSKIL
MEDAN

2016

LEMBARAN PENGESAHAN

PENCARIAN *MAXIMUM CLIQUE* MENGGUNAKAN ALGORITMA *IMPROVED ANT COLONY OPTIMIZATION* DAN *PARTICLE SWARM OPTIMIZATION* PADA JARINGAN SOSIAL

TUGAS AKHIR

Diajukan untuk Melengkapi Persyaratan Guna
Mendapatkan Gelar Sarjana Strata Satu
Program Studi Teknik Informatika

Oleh :

ALEXANDER YAP
NIM. 121110341

JEFFRY OWEN
NIM. 121110227

Disetujui Oleh:

Dosen Pembimbing I,	Dosen Pembimbing II,
_____	_____

Medan, 11 Juli 2016
Diketahui dan Disahkan Oleh:

Ketua Program Studi
Teknik Informatika,

_____

ABSTRAK

Pada teori kompleksitas komputasi, pencarian maksimum *clique* merupakan permasalahan yang dikategorikan *Nondeterministic-Polinomial* (NP)-*Hard* dan digunakan sebagai alat untuk menganalisis interaksi antar aktor pada jaringan sosial. Untuk menyelesaikan permasalahan ini, pendekatan optimalisasi dapat diaplikasikan seperti *Ant Colony Optimization* (ACO), *Improved Ant Colony Optimization* (IACO), dan *Ant Colony Optimization* ditambah *Particle Swarm Optimization* (ACOPSO). Agar mendapatkan hasil yang lebih baik, algoritma IACO dengan penemuan hasil yang lebih tinggi dari ACO digabungkan dengan ACOPSO dengan waktu pencarian yang rendah dibandingkan dengan ACO menjadi *Improved Ant Colony Optimization* ditambah *Particle Swarm Optimization* (IACOPSO). Pengujian dilakukan pada graf jaringan sosial dan DIMACS dimana pada graf DIMACS, dilakukan pembagian menjadi 3 bagian berdasarkan jumlah simpul. Hasil penelitian menunjukkan algoritma IACOPSO mampu memberikan hasil yang lebih baik dengan waktu yang relatif lebih singkat.

Kata Kunci : *Improved Ant Colony Optimization*, Maksimum *clique,* Jaringan Sosial, ACO, PSO

KATA PENGANTAR

Ucapan syukur penulis panjatkan Tuhan Yang Maha Esa karena berkat Rahmat dan Karunia-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul "Pencarian *Maximum Clique* Menggunakan Algoritma *Improved Ant Colony Optimization* dan *Particle Swarm Optimization* Pada Jaringan Sosial".

Penulisan Tugas Akhir ini diajukan untuk memenuhi salah satu syarat kelulusan dalam jenjang perkuliahan Strata I Sekolah Tinggi Manajemen Informatika Dan Komputer.

Dalam kesempatan ini penulis ingin mengucapkan terima kasih kepada :

1. Bapak Arwin Halim, S.Kom., M.Kom., selaku Pembimbing I yang telah membimbing penulis mengerjakan Tugas Akhir ini.
2. Bapak Irpan Adiputra Pardosi, S.Kom., M.TI, selaku Pembimbing II yang telah membimbing penulis mengerjakan Tugas Akhir ini.
3. Bapak Dr. Mimpin Ginting, M.S., selaku Ketua STMIK Mikroskil Medan
4. Bapak Djoni, S.Kom., M.T.I., selaku Wakil Ketua I STMIK Mikroskil Medan.
5. Bapak Hardy, S.Kom., M.Sc., selaku Ketua Program Studi Teknik Informatika STMIK Mikroskil Medan.
6. Bapak dan Ibu Dosen yang telah mendidik dan membimbing penulis dalam mengerjakan Tugas Akhir ini.
7. Orang tua dan teman-teman kami yang selama ini telah men-*support* penulis tetap semangat untuk menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa penelitian Tugas Akhir ini masih jauh dari sempurna sehingga penulis membutuhkan kritik dan saran yang bersifat membangun untuk kemajuan pendidikan di masa yang akan datang.

Akhir kata, semoga penelitian Tugas Akhir ini memberikan manfaat, khususnya bagi penulis dan umumnya bagi semua orang dalam rangka menambah wawasan pengetahuan.

Medan, Agustus 2016

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Jaringan sosial merupakan suatu konsep yang digunakan untuk merepresentasikan dan menganalisis data yang berhubungan dengan interaksi sosial (Baglioni et al., 2014). Terkadang interaksi sosial dapat membentuk graf sempurna yang dikenal dengan *clique*[1]. Terdapat beberapa permasalahan yang berkaitan dengan *clique* salah satunya adalah pencarian maksimum *clique*[2]. Pencarian maksimum *clique* merupakan permasalahan yang dikategorikan kedalam *Nondeterministic-Polynomial (NP)-Hard* dalam teori kompleksitas komputasi (Solnon & Fenet, 2006), karena waktu yang dibutuhkan untuk mendapatkan hasil akan meningkat secara eksponensial berdasarkan jumlah simpul yang ada pada graf (Bomze et al., 1999).

Terdapat beberapa permasalahan yang mengimplementasikan pencarian maksimum *clique* seperti menganalisis pola pada lalu lintas telekomunikasi, mendiagnosis kesalahan, pendeteksi pola (Bomze et al., 1999), dan analisis jaringan sosial (Yan et al., 2011) seperti mendeteksi komunitas dan analisis jaringan teroris pada penyerangan *World Trade Center* (Krebs, 2002). Tujuan dasar dari menganalisis jaringan sosial adalah memberikan pemahaman tentang hubungan sosial antar aktor dan mencegah terjadinya hal yang tidak diinginkan (Pattilo et al., 2012).

Banyak penelitian dilakukan untuk mendapatkan algoritma paling efisien dalam menemukan maksimum *clique* terutama dari segi waktu. Beberapa algoritma yang telah diuji yaitu algoritma *Ant Colony Optimization* (Solnon & Fenet, 2006), *Improved Ant Colony Optimization* (Xu et al., 2007), *Reactive local search* (RLS) (Battiti & Protasi, 1995), algoritma *Intelligent Waterdrop* (Taani & Nemrawi, 2012), dan pengabungan antara algoritma *Ant Colony Optimization* (ACO) dengan *Particle Swarm Optimization* (PSO) atau dikenal dengan algoritma

[1]Setiap aktor memiliki hubungan satu dengan yang lainnya. (Easley & Kleinberg, 2010).
[2]*clique* dengan jumlah simpul terbanyak.

hibrida (Pouri et al., 2012). Hasil penelitian menunjukkan algoritma *Ant Colony Optimization* mampu memberikan hasil yang baik dibandingkan dengan algoritma lainnya. Oleh karena itu, muncul penelitian yang bertujuan untuk meningkatkan kinerja dari *Ant Colony Optimization* dalam mencari maksimum *clique* seperti yang dilakukan oleh Xu dan Pouri.

Penelitian Xu dkk. (2007), memperkenalkan *Improved Ant Colony Optimization* yang merubah algoritma *Ant Colony Optimization* dan ditambahkan dengan metode *simulated annealing* dalam penentuan jalur. Penelitian Pouri dkk. (2012), memperkenalkan algoritma hibrida dengan menggabungkan algoritma *Ant Colony Optimization dan Particle Swarm Optimization* pada penambahan *pheromone.* Hasil penelitian menunjukkan algoritma *Improved Ant Colony Optimization* memberikan hasil yang lebih baik dari *Ant Colony Optimization* (Xu et al., 2007) dan algoritma hibrida antara *Ant Colony Optimization* dan *Particle Swarm Optimization* memberikan hasil pencarian dengan waktu yang lebih cepat dibandingkan dengan *Ant Colony Optimization* (Pouri et al., 2012). Dalam tugas akhir ini, dibangun program yang menggabungkan algoritma *Improved ant colony Optimization* dan *Particle Swarm Optimization (IACOPSO)* untuk menemukan maksimum *clique* pada graf jaringan sosial.

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah di atas, dapat ditarik rumusan permasalahan yang akan dibahas yaitu bagaimana kemampuan *Improved Ant Colony Optimization* ditambah *Particle Swarm Optimization* dibandingkan dengan algoritma sebelumnya dalam mencari maksimum *clique* pada jaringan sosial.

1.3 Tujuan dan Manfaat

Adapun tujuan dari tugas akhir ini adalah membangun aplikasi untuk mencari maksimum *clique* dengan menggunakan algoritma *Improved Ant Colony Optimization* digabung dengan *Particle Swarm Optimization.*

Manfaatnya yaitu:

1. Sebagai pembelajaran lebih lanjut mengenai maksimum *clique*.
2. Sebagai bahan referensi untuk penelitian selanjutnya.
3. Membantu dalam analisis jaringan sosial yang berhubungan dengan maksimum *clique*.

1.4 Batasan Masalah

Batasan masalah dari tugas akhir ini adalah:

1. Graf yang digunakan adalah graf tak berarah.
2. Jumlah simpul yang digunakan maksimal sebanyak 4.500.
3. Penempatan *pheromone* dilakukan pada sisi.
4. *Dataset* yang digunakan terdiri dari:
   a. Graf jaringan sosial dimana graf yang akan diuji terdiri dari:
      i. *Zachary's Karate Club*[3] dengan 34 simpul dan 78 sisi.
      ii. *Social Network of dolphins*[1] dengan 62 simpul dan 159 sisi.
      iii. *Facebook Social Circle*[4] dengan 4.039 simpul dan 88.234 sisi.
      iv. *Erdos Collaboration Network 991*[5] dengan 492 simpul dan 1417 sisi.
      v. *Erdos Collaboration Network 971* dengan 472 simpul dan 1314 sisi.
      vi. *Co-Authorship of Science in Network Theory ant Experiment* dengan 1589 dan 1190 sisi.
      vii. *Email Interchange Network, University of Rovia I Virgili, Tarragona* dengan 1133 simpul dan 5451 sisi.
   b. Graf dari DIMACS[6] dimana graf yang akan diuji berkategori graf kecil, graf, dan graf besar. Pembagian kategori akan dijelaskan pada Bab 3 Sub-bab 3.2.

[3] http://www-personal.umich.edu/~mejn/netdata/
[4] http://snap.stanford.edu/data/
[5] http://vlado.fmf.uni-lj.si/pub/networks/data/
[6] http://iridia.ulb.ac.be/~fmascia/maximum_clique/DIMACS-benchmark

1.5 Metodologi Penelitian

Dalam pelaksanaan tugas akhir ini, langkah-langkahnya sebagai berikut:

1. Pengumpulan referensi yang berhubungan dengan pencarian maksimum *clique*, penjelasan tentang algoritma *Ant Colony Optimization* dan *Particle Swarm Optimization,* serta referensi-referensi lain yang berhubungan dengan tugas akhir ini.
2. Pengembangan sistem yang mengacu pada metode *waterfall*. Adapun tahapan-tahapan dalam metode *waterfall* adalah:

   a. Analisis

      Pada tahap ini mencakup analisis proses, pemodelan sistem, identifikasi persyaratan fungsional dan non-fungsional dari sistem yang akan dirancang. Analisis proses sistem akan digambarkan dalam bentuk UML *activity diagram* untuk algoritma *Improved Ant Colony Optimization* digabung dengan *Particle Swarm Optimization*. Analisis kebutuhan fungsional akan digambarkan dengan diagram *use case*.

   b. Perancangan

      Pada tahap ini akan menerjemahkan syarat kebutuhan fungsional yang sebelumnya telah digambarkan dengan diagram *use case* kedalam perancangan perangkat lunak. Tahapan proses ini berfokus pada tampilan aplikasi yang dibangun. Rancangan awal akan digambarkan dengan *mockup* kemudian rancangan akhir akan dibangun dengan *windows form.*

   c. Pengkodean

      Pada tahap ini akan dilakukan implementasi dari analisis dan desain dengan menggunakan bahasa pemrograman C#.

   d. Pengujian

      Proses pengujian akan dilakukan sebagai berikut:

      i. Pengujian awal dilakukan dengan mencari nilai rho (tingkat evaporasi) untuk algoritma *Improved Ant colony Optimization* ditambah *Particle Swarm Optimization* (IACOPSO) dimana

nilai yang diuji terdiri dari 0.1, 0.3, 0.5, 0.7, 0.9 dan 1 ( ). *Dataset* yang digunakan adalah *dataset* jaringan sosial dan *dataset* DIMACS [7]. Untuk mempermudah pengujian dan analisis hasil maka graf DIMACS akan dibagi menjadi 3 kelompok yaitu graf kecil, graf sedang, dan graf besar. Tabel dari kelompok *dataset* DIMACS dapat dilihat pada bab 3 *subbab* 3.2 Pengumpulan data.

ii. Setelah menemukan nilai rho yang sesuai, maka parameter rho akan ditetapkan untuk algoritma *Improved Ant Colony Optimization* ditambah *Particle Swarm Optimization* (IACOPSO) dalam mencari maksimum *clique* pada *dataset* jaringan sosial dan DIMACS. Sedangkan parameter lain akan disamakan dengan penelitian sebelumnya.

iii. Untuk mengetahui performansi dari algoritma IACOPSO, maka hasil dari algoritma IACOPSO akan dibandingkan dengan hasil dari algoritma ACO, *Improved* ACO (IACO), dan ACO gabung PSO (ACOPSO) dengan menggunakan *dataset* yang sama.

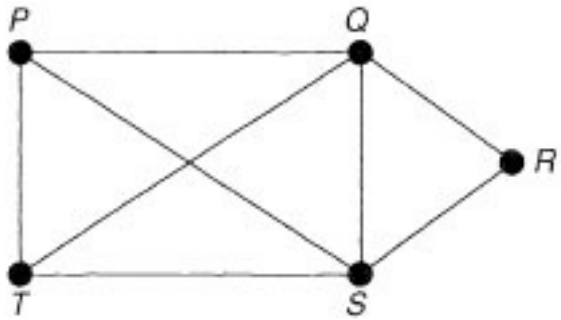
3. Menarik kesimpulan bedasarkan hasil pengujian terhadap program dan menyusun laporan tugas akhir.

[7] Center for Discrete Mathematics and Theoretical Computer Science.

BAB II
TINJAUAN PUSTAKA

2.1 Teori Graf

Teori graf menjadi alat matematika yang sangat penting dalam berbagai bidang (Wilson, 1996) dan salah satu contoh penggunaanya yaitu skema jalan dimana kota dapat diibaratkan sebagai simpul sedangkan jalan dapat diibaratkan sebagai sisi. Sisi akan menghubungkan kota yang satu dengan kota yang lain sehingga membentuk suatu relasi antar kota.



Gambar 2.1 Graf
Sumber : (Wilson, 1996).

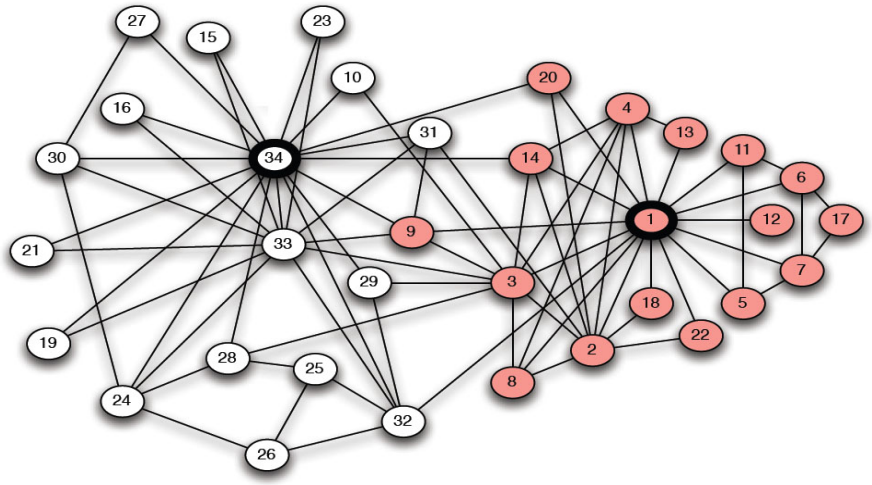
Titik P,Q,R,S,T pada Gambar 2.1 dikenal dengan simpul sedangkan garis yang menghubungkan antar simpul disebut sebagai sisi. Derajat atau *degree* dari suatu simpul menyatakan jumlah sisi yang berhubungan dengan simpul tersebut. Misalkan simpul Q memiliki derajat 4, simpul P memiliki derajat 3, dan simpul R memiliki derajat 2 (Wilson, 1996).

2.2 Jaringan

Jaringan merupakan kumpulan dari objek yang memiliki keterkaitan satu sama lain (Easley & Kleinberg, 2010). Perkembangan teknologi dan komunikasi global telah meningkatkan kompleksitas dari jaringan terutama jaringan sosial. Jaringan dapat digunakan untuk mempelajari dan memahami hubungan suatu objek dan pengaruhnya terhadap objek lainnya.

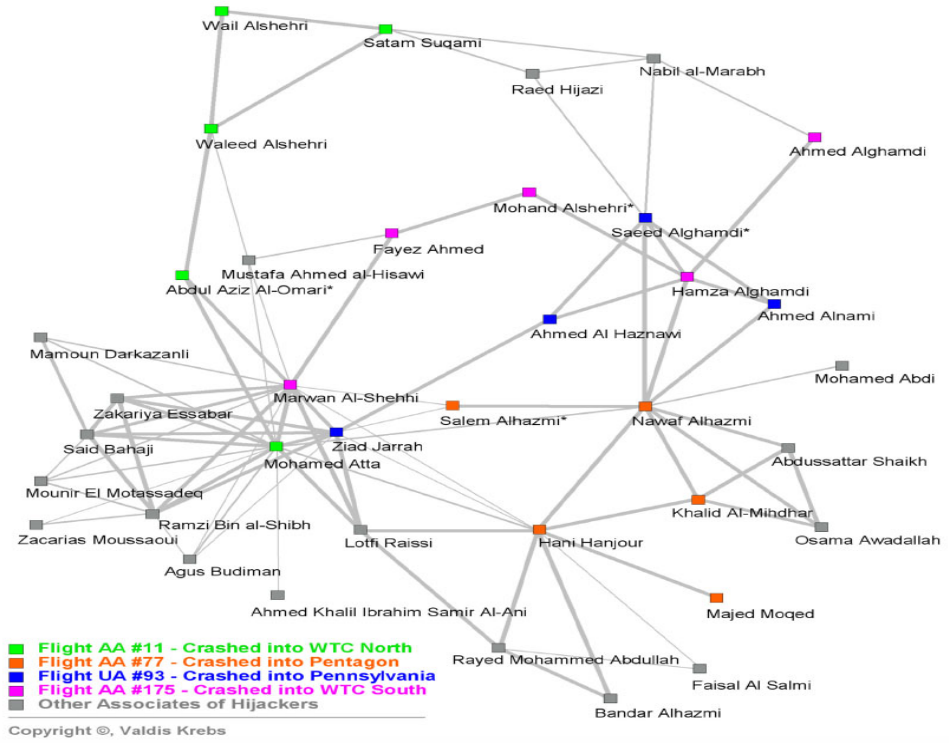
2.2.1 Jaringan Sosial

Manusia sebagai mahluk sosial akan bersosialisasi dengan manusia lainnya. Tingkat sosialisasi manusia berbeda-beda, ada yang memiliki tingkat sosialisasi yang tinggi dan ada yang rendah (Christakis & Fowler, 2009). Untuk dapat memahami dan mempelajari lebih lanjut tentang hubungan antar manusia, maka hubungan tersebut digambarkan kedalam bentuk jaringan yang dikenal dengan jaringan sosial. Jaringan sosial merupakan suatu konsep yang digunakan untuk merepresentasikan data yang berhubungan dengan interaksi sosial (Baglioni et al., 2014). Jaringan sosial kemudian dianalisis untuk mendapatkan informasi-informasi penting yang berhubungan dengan kebiasaan sosial dari setiap aktor (Wasserman & Faust, 1994). Semakin tinggi tingkat sosialisasi seseorang, maka orang tersebut akan tersentralisasi pada jaringan sosial yang dikenal dengan istilah *central* (Christakis & Fowler, 2009).



Gambar 2.2 Jaringan Pertemanan Antara 34 Orang pada Klub Karate
Sumber: (Easley & Kleinberg, 2010)

Gambar 2.2 merupakan bentuk jaringan sosial pertemanan pada klub karate di sebuah universitas. Jaringan tersebut terbagi menjadi 2 kelompok kecil dimana simpul nomor 1 dan 34 sebagai *central*. Hal ini dapat dilihat dari hubungan antar simpul yang terkoneksi dengan kelompoknya sendiri dan hanya sedikit dari simpul yang berhubungan antar kelompok. Ini menunjukkan adanya konflik pada jaringan tersebut (Easley & Kleinberg, 2010).



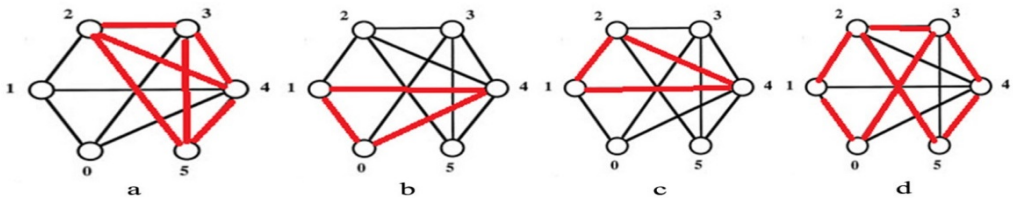
Gambar 2.3 Jaringan Sosial *Hijacker's Network Neighborhood*
Sumber: (Krebs, 2002).

Gambar 2.3 memperlihatkan jaringan terorisme yang dibentuk oleh Krebs bedasarkan penyerangan *World Trade Center* pada 11 September 2001 (Krebs, 2002). Terdapat 19 teroris yang terlibat dalam penyerangan dan beberapa aktor yang memiliki kemungkinan membantu penyerangan tersebut (Krebs, 2002).

2.2.1.1 Clique

Pada jaringan sosial, *Clique* atau graf sempurna diartikan sebagai sekelompok orang yang saling mengenal satu sama lain (Easley & Kleinberg, 2010). *Clique* juga dapat diartikan sebagai hubungan antara sekelompok aktor yang memiliki tujuan yang sama (Pattilo et al., 2012).

Salah satu masalah berkaitan dengan *clique* adalah mencari *clique* maksimum dan dikategorikan sebagai permasalahan NP-*Hard.* Ide utama dalam permasalahan maksimum *clique* adalah mencari *clique* dengan jumlah simpul paling besar dalam graf G (Solnon & Fenet, 2006).



Gambar 2.4 Contoh *Clique* pada Graf
Sumber: (Roberts & Hennesy, 2003)

Graf pada Gambar 2.4(a) termasuk *clique* maksimum dimana setiap simpul 2,3,4,5 saling terhubung satu sama lain dengan ukuran 4. Gambar 2.3(b) dan 2.3(c) termasuk *clique* tetapi tidak termasuk sebagai *clique* maksimum. Sedangkan pada gambar 2.3(d) bukan sebuah *clique* karena simpul 0 dan 5 tidak saling berhubungan.

2.3 Artificial Intelligence

Artificial intelligence (AI) merupakan salah satu pembelajaran yang mempelajari dan membentuk teknologi yang dapat mengambil tindakan dengan cerdas (Brownlee, 2011). Menurut Russel dan Norvig, AI didefinisikan menjadi empat kategori (Russel & Norvig, 1995)

1. Sistem yang berpikir layaknya manusia
2. Sistem yang bertindak layaknya manusia
3. Sistem yang berpikir rasional
4. Sistem yang bertindak secara rasional

Terdapat dua jenis pencarian yang dikenal pada AI dalam mencari solusi yaitu pencarian secara heuristik dan pencarian secara metaheuristik.

2.3.1 Pencarian Heuristik

Menurut Luger (2005), Penyelesaian AI dilakukan secara heuristik dengan dua situasi dasar yaitu:

1. Permasalahan tidak memiliki solusi yang tepat karena adanya ambiguitas pada data. Contohnya pada diagnosis medis. Sebuah penyakit terkadang memiliki banyak gejala sehingga dokter menggunakan pendekatan heuristik untuk mendapatkan solusi yang mendekati kebenaran.

2. Permasalahan memiliki solusi yang tepat, akan tetapi membutuhkan waktu lama untuk mendapatkan solusi. Contohnya pada permasalahan catur.

Pencarian Heuristik biasanya bersifat *problem dependent* dimana pencarian heuristik menggunakan pengetahuan khusus dari suatu masalah untuk dapat menyelesaikannya (Russel & Norvig, 1995). Pencarian heuristik juga sering memberikan hasil yang *sub-optimal* atau bahkan gagal dalam menemukan solusi (Luger, 2005) dikarenakan pencarian heuristik tidak mencari semua kemungkinan yang ada untuk mendapatkan hasil yang lebih baik (Li et al., 2007). Untuk mengatasi masalah dari pencarian heuristik, muncul pencarian metaheuristik.

2.3.2 Pencarian Metaheuristik

Dikarenakan pencarian heuristik sering terjebak pada hasil yang bersifat lokal, maka dilakukan pengembangan lanjutan dari pendekatan heuristik untuk meningkatkan hasil yang didapatkan (Yang, 2014). Peningkatan kinerja dari pencarian heuristik dilakukan dengan menggabungkan beberapa metode dari pendekatan heuristik dan terdapat beberapa sifat dari pencarian metaheuristik yaitu:

1. Pencarian metaheuristik merupakan strategi-strategi yang memandu pencarian.
2. Tujuan pencarian metaheuristik adalah mengefisiensikan pencarian untuk mendapatkan solusi yang optimal atau mendekati optimal.
3. Teknik pada algoritma metaheuristik dapat berupa pencarian simpel sampai proses pembelajaran yang rumit.
4. Algoritma metaheuristik bersifat *non-deterministic* dimana inputan sama dapat memberikan jalan yang berbeda.
5. Terdapat penambahan mekanisme pada algoritma untuk menghindari terjebaknya algoritma pada hasil lokal optimal. (Brownlee, 2011).

Selain meningkatkan hasil, penggabungan dari beberapa pendekatan heuristik bertujuan agar dapat diaplikasikan kedalam banyak permasalahan yang berbeda dengan perubahan struktur algoritma yang lebih sedikit (Dorigo & Stutzle, 2004). Setiap agen dari algoritma metaheuristik menyimpan hasil yang

didapatkan dalam penelusuran dan akan dibandingkan satu sama lain untuk menghasilkan hasil yang terbaik (Dorigo & Stutzle, 2004).

2.3.2.1 Particle Swarm Optimization

Particle Swarm Optimalization (PSO) merupakan algoritma yang diciptakan bedasarkan observasi tingkah laku mahluk hidup dimana mahluk hidup tersebut membentuk suatu koloni atau kelompok untuk mencari makanan ataupun bertahan hidup seperti kelompok burung ataupun sekawanan ikan (Kennedy & Eberhart, 1995). Pada dasarnya, sekelompok burung akan bertengger di atas pohon ataupun pada kabel telepon, namun alasan utama sekelompok burung bertengger karena adanya makanan. Setiap anggota akan mengingat hasil terbaik dan posisi dimana hasil terbaik itu dihasilkan. Hasil terbaik dinamakan *pbest[]* sedangkan posisinya dikenal dengan *pbestx[]* dan *pbesty[]*. Setiap agen akan mengetahui posisi terbaik secara global yang pernah ditemukan oleh salah satu anggota dalam kelompok yang dikenal dengan *gbest*. Jadi, *pbestx[gbest]* merupakan posisi X terbaik yang pernah ditemukan dan *pbesty[gbest*] merupakan posisi Y terbaik yang pernah ditemukan oleh kelompok burung tersebut (Kennedy & Eberhart, 1995).

Dalam algoritma, setiap solusi yang dikenal dengan istilah *particle* akan dibentuk secara acak pada awal pengeksekusian dan akan terus diperbaharui pada tiap langkah untuk mendapatkan solusi yang optimal (Pouri et al., 2012). Kecepatan dan lokasi dari tiap *particle* akan diperbaharui menggunakan Persamaan 1:

(1)

Dimana:

pbest = Posisi terbaik yang pernah dicapai oleh partikel.

gbest = Posisi terbaik yang pernah dicapai oleh kumpulan partikel.

pada iterasi ke t

= Nilai random dengan jarak (0,1)

= Parameter pembelajaran yang diasumsikan sama ().

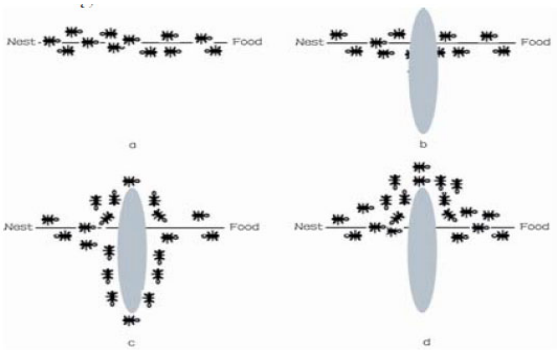
= Berat inersia yang dianggap bernilai konstan atau acak.

2.3.2.2 Ant Colony Optimization

Algoritma semut atau dikenal dengan algoritma *Ant Colony Optimization* (ACO) merupakan algoritma pencarian metaheuristik yang dibuat bedasarkan observasi dari tingkah laku koloni semut dan digunakan untuk menyelesaikan masalah optimalisasi. (Dorigo & Stutzle, 2004). Di dunia nyata, koloni semut akan keluar dari sarang untuk mencari makanan. Rute yang dilalui semut-semut untuk mencari makanan akan dilakukan secara acak dan pada saat semut tersebut menemukan makanan dan kembali ke sarang, maka semut tersebut akan meninggalkan sebuah tanda yang dikenal dengan *pheromone* (Dorigo & Stutzle, 2004).

Jumlah *pheromone* yang ditinggalkan tergantung pada kualitas dan ukuran dari makanan yang ditemukan. *Pheromone* yang ditinggalkan tidak akan bertahan lama dan akan menguap sehingga jumlah *pheromone* paling sedikit akan hilang sedangkan yang paling banyak akan bertahan. Sisa *pheromone* yang tertinggal akan menarik semut-semut lain untuk mengikuti jalur tersebut (Pouri et al., 2012).

Ide awal algoritma *Ant Colony Optimization* yaitu untuk mencari jalan dengan nilai minimum pada suatu graf (Dorigo & Stutzle, 2004). Hal ini diketahui melalui percobaan kedua yang dilakukan oleh Deneubourg dkk. bedasarkan pengamatan semut *Iridomyrmex humilis* (Deneubourg et al., 1989). Dari percobaan kedua yang dilakukan, dapat dilihat bahwa, koloni semut *Iridomyrmex humilis* akan memilih jalan terpendek untuk mencapai tujuan (Deneubourg et al., 1989) karena jumlah *pheromone* akan terkumpul lebih cepat pada jalan yang pendek dibanding jalan yang panjang.



Gambar 2.5 Pemilihan Rute Terpendek oleh Koloni Semut
Sumber : (Xu et al., 2007).

Cara kerja algoritma *Ant Colony Optimization* berawal dari menentukan solusi berbeda secara *random*, kemudian solusi tersebut akan ditingkatkan dengan mengupdate *pheromone*. Berdasarkan masalah pencarian *clique*, *pheromone* dapat diletakkan pada titik atau sisi pada graf. Perpindahan semut pada titik i ke j dapat dirumuskan dengan Persamaan 3:

(3)

Dimana:

= Probabilitas perpindahan dari titik i ke titik j oleh semut k.

= Jumlah *pheromone* yang ada pada sisi dari i dan j yang akan dilalui.

= Faktor *pheromone*.

Pemilihan probabilitas jalan dilakukan secara *incremental* dimana saat simpul pertama ditentukan secara acak () dan kandidat yang berhubungan dengan (), maka faktor *pheromone* dari simpul antara dan ()) diinisialisasikan dengan . Kemudian, saat setiap simpul dimasukan kedalam *clique* () dan berhubungan dengan , maka) akan di*increment* dengan (Solnon & Fenet, 2006).

Metode untuk melakukan update *pheromone* dapat dituliskan dengan Persamaan 4 dan Persamaan 5:

(4)

(5)

Dimana:

= Tingkat penguapan dari *Pheromone*(antara 0 dan 1).

= Jumlah *pheromone* baru pada sisi antara titik i dan j.

= Jumlah *pheromone* saat ini pada sisi antara titik i dan j.

= Jumlah *pheromone* yang ditambahkan untuk solusi yang diinginkan.

Tahap awal dalam melakukan update *pheromone* yaitu melakukan evaporasi pada setiap *pheromone* yang ada menggunakan Persamaan 4. Kemudian penambahan *pheromone* dilakukan pada solusi yang diinginkan menggunakan Persamaan 5.

Jumlah *pheromone* yang ditambahkan () dapat dihitung sesuai Persamaan 6.

(6)

Dimana:

G_{best} = Hasil yang paling baik yang pernah ditemukan

Best tour = Hasil terbaik yang ditemukan dalam iterasi.

Algoritma *Ant Colony Optimization* yang dibuat oleh Fennet dan Solnon yang disebut dengan *Ant-Clique* untuk menyelesaikan permasalahan *clique* maksimum dapat dilihat pada Gambar 2.6.

```
procedure Ant-Clique
Initialize pheromone trails
repeat
for k in 1..nb Ants do: construct a clique Ck
Update pheromone trails w.r.t. {C1, . . . , CnbAnts}
until max cycles reached or optimal solution found.
```

Gambar 2.6 Algoritma *Ant-Clique*
Sumber: (Solnon & Fenet, 2006)

Tahap awal algoritma *Ant-Clique* dimulai dari inisialisasi jejak *pheromone* pada sisi. Kemudian setiap semut akan mulai membentuk *clique* dengan cara memilih simpul awal secara acak, kemudian memilih simpul lainnya yang akan dimasukkan kedalam C_k. Algoritma untuk membentuk *clique* dapat dilihat pada Gambar 2.7 (Solnon & Fenet, 2006).

```
Choose randomly a first vertex vf ∈ V
C ← {vf }
Candidates ← {vi/(vf, vi) ∈ E}
while Candidates _= ∅ do
    Choose a vertex vi∈ Candidates with probability
     p(vi) =
C ← C U{vi}
Candidates ← Candidates ∩ {vj/(vi, vj) ∈ E}
end while
return C
```

Gambar 2.7 Algorimta *Ant-Clique* dalam Membentuk *Clique*
Sumber : (Solnon & Fenet, 2006)

Sesudah tiap semut selesai membentuk *clique,* maka jejak *pheromone* akan diperbaharui sesuai dengan Persamaan 4 dan Persamaan 5. *Pheromone* akan dikurangi untuk mensimulasikan penguapan *pheromone* di dunia nyata. Kemudian *pheromone* akan ditambahkan berdasarkan Persamaan 6. Nilai dari alpha (dan tingkat penguapan atau evaporasi atau rho (merupakan parameter yang paling penting dalam menemukan hasil oleh algoritma *Ant Colony Optimization.* Nilai alpha berpengaruh terhadap sensitivitas dari semut terhadap jejak *pheromone* dan tingkat penguapan berpengaruh terhadap kecepatan penguapan dari *pheromone*.

2.4 Improved Ant Colony Optimization

Menurut Xu dkk. (2007), simpul yang dimasukkan ke C_k pada algoritma *Ant Colony Optimization* Fennet dan Solnon hanya bergantung pada jejak *pheromone*. Akan tetapi, jumlah derajat pada simpul dapat menjadi pertimbangan

dalam memilih simpul yang akan dilalui oleh semut sehingga dapat meningkatkan peformansi dari algoritma *Ant Colony Optimization*. *Improved Ant Colony Optimization* (IACO) menggunakan algoritma *ant-clique* dari Fennet dan Solnon dan dilakukan perubahan dalam pemilihan simpul kandidat dimana pemilihannya dilakukan dengan Persamaan 7. IACO juga menambahkan metode *annealing* untuk meningkatkan hasil yang didapat. Proses *annealing* dapat dilihat pada Persamaan 8.

(7)

Dimana:

Degree = Derajat ke pada simpul (degree(i) = dimana = 1 jika . Jika tidak, = 0).

EdgeNum = Jumlah sisi pada graf G (EdgeNum = dimana = 1 jika . Jika tidak, = 0).

T(t) = Variabel temperatur (sebagai faktor penurunan eksponensial agar semut dapat mencapai tingkat stabil).

Pembaharuan T(t) dilakukan dengan Persamaan 8

(8)

Dimana:

T(t+1) = Pembaharuan dari T(t).

= Faktor pengurangan.

Dengan perubahan pada Persamaan 7, maka probabilitas pemilihan simpul kandidat akan berfokus pada jumlah derajat yang besar pada graf sehingga dapat membantu semut dalam menemukan solusi optimal dengan lebih baik dan cepat. Persamaan 8 bertujuan agar semut dapat mencapai kondisi stabil.

Untuk meningkatkan kecepatan dari pencarian, IACO merubah parameter alpha dan rho yang statis dari penelitian sebelumnya dengan parameter yang dinamis dimana perubahan nilai alpha dan rho dapat dilihat pada Persamaan 9 dan perubahan nilai rho dapat dilihat pada Persamaan 10.

(9)

(10)

Dimana:

t = Jumlah iterasi

= Konstanta kecil yang bernilai positif

2.5 *Ant Colony Optimization* Ditambah *Particle Swarm Optimization*

Menurut Pouri dkk. (2012), kelemahan dari penambahan pheromone dari pendekatan heuristik sebelumnya adalah terjadi peningkatan dalam jumlah perhitungan pada perhitungan penambahan *pheromone* (seperti pada Persamaan 6). Untuk mengurangi peningkatan perhitungan *pheromone*, maka dilakukan penggabungan antara ACO dan PSO untuk mencari *clique* maksimum pada graf. ACO digunakan untuk menelusuri graf dan memberi *pheromone* ,sedangkan PSO digunakan untuk menghitung peningkatan *pheromone*. Prosedur ini akan terus berulang sampai menemukan *clique* maksimum. pada *pheromone* yang telah diperkuat untuk *clique* yang diinginkan akan dihitung dengan algoritma PSO menggunakan Persamaan 11:

(11)

Dimana:

= Jumlah penambahan *pheromone* saat ini pada simpul i dan j.

= Jumlah penambahan *pheromone* baru pada simpul i dan j dan pada iterasi ke t+1.

= Nilai penambahan pada iterasi t.

Nilai dapat ditentukan menggunakan Persamaan 12:

(12)

Dimana:

= Nilai baru dari pada iterasi ke t+1.

= Dua nilai acak diantara 0 sampai 1.

= Parameter pembelajaran. ()

= Jumlah simpul *clique* terbaik sekarang dan jumlah simpul *clique* terbaik yang telah ditemukan sampai saat ini.

2.6 Improved Ant Colony Optimization ditambah Particle Swarm Optimization

Dari penelitian Xu, dkk. (2007), *Improved Ant Colony Optimization* dapat memberikan hasil yang baik dibandingkan dengan *Ant Colony Optimization*. Sedangkan pada penelitian Pouri, dkk. (2012), *Ant Colony Optimization*.yang ditambah dengan *Particle Swarm Optimization* pada proses penambahan *pheromone* dapat memberikan hasil yang lebih baik dengan waktu yang lebih singkat dibandingkan dengan *Ant Colony Optimization*. Maka *Improved Ant Colony Optimization* akan digabung dengan *Ant Colony Optimization* ditambah *Particle Swarm Optimization* agar dapat meningkatkan hasil pencarian. Pemilihan kandidat menggunakan Persamaan 7 dan 8. Penggunaan parameter alpha dan rho dilakukan secara dinamis dengan menggunkan Persamaan 9 dan 10, sedangkan penambahan *pheromone* pada solusi yang terbaik akan ditambahkan menggunakan Persamaan 11 dan 12. Untuk Persamaan 10, penentuan tingkat evaporasi akan disesuaikan dengan hasil pengujian yang akan dilakukan pada Bab 4.

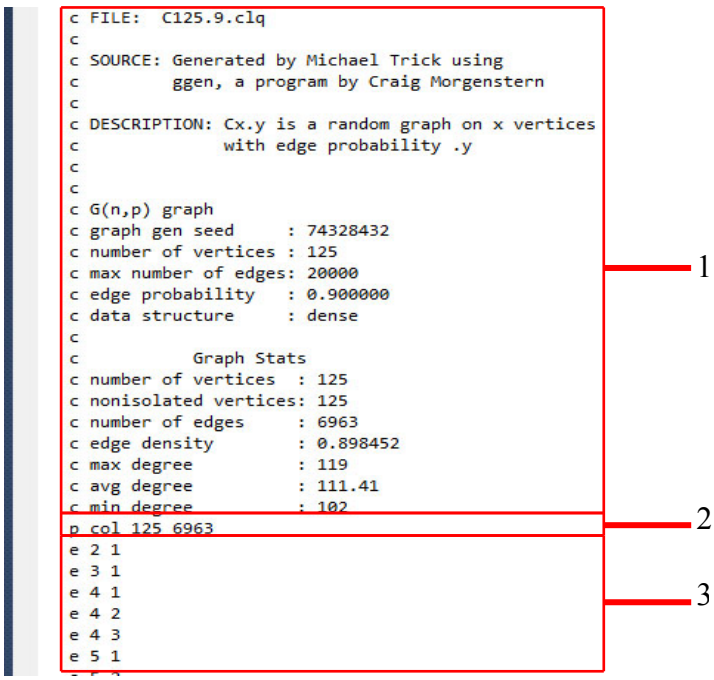
BAB III
METODOLOGI PENELITIAN

3.1 Pengumpulan Referensi

Referensi awal yang dikumpulkan adalah referensi-referensi mengenai Permasalahan maksimum *clique* dan referensi penelitian algoritma-algoritma yang digunakan untuk menyelesaikan permasalahan maksimum *clique*. Dari hasil pengumpulan referensi didapat dasar utama untuk menggabungkan algoritma *Improved Ant Colony Optimization,* dan *Ant Colony Optimization* ditambah *Particle Swarm Optimization* menjadi algoritma *Improved Ant Colony Optimization* ditambah *Particle Swarm Optimization*.

3.2 Pengumpulan *Dataset*

Terdapat dua jenis *dataset* yang digunakan yaitu *dataset* jaringan sosial dan *dataset* DIMACS. Format data yang digunakan mengikuti format dari DIMACS dimana terdapat kolom komentar, inisialisasi permasalahan dan hubungan antar simpul. Bentuk format *dataset* yang digunakan dapat dilihat pada Gambar 3.1



Gambar 3.1 Format *Dataset*

Keterangan Gambar 3.1:

1. Kolom komentar : Biasa ditandai dengan c, %, atau *. Berisi penjelasan tentang *dataset*.
2. Kolom inisialisasi permasalahan : berisi jumlah simpul dan jumlah sisi pada *dataset*.
3. Kolom hubungan simpul : Berisi hubungan antar simpul. Pada Gambar 3.1, dapat dilihat simpul 2 terhubung dengan simpul 1. Simpul 3 terhubung dengan simpul 1.

Untuk *dataset* jaringan sosial yang digunakan dapat dilihat pada Tabel 3.1.

Tabel 3.1 Jaringan Sosial

Dataset	***Max Clique***	**Simpul**	**Sisi**
Zachary's karate club	-	34	78
Social network of dolphins	-	62	159
Facebook Social circle	-	4039	88234
Erdos Collaboration Network 991	-	492	1417
Erdos Collaboration Network 971	-	472	1314
Co-Authorship of Science in Network Theory ant Experiment	-	1589	1190
Email Interchange Network,University of Rovia I Virgili, Tarragona	-	1133	5451

Untuk *dataset* DIMACS, dilakukan pembagian menjadi 3 bagian berdasarkan jumlah simpul. Hal ini bertujuan untuk mempermudah pengujian dan analisis. Pembagian *dataset* DIMACS dapat dilihat pada Tabel 3.2, 3.3, dan 3.4.

Tabel 3.2 DIMACS Dengan Graf Kecil

Dataset	***Max Clique***	**Simpul**	**Sisi**
p_hat300-1	8,00	300	10933
brock200_2	12,00	200	9876
hamming8-4	16,00	256	20864
brock200_4	17,00	200	13089
brock400_2	29,00	400	59786
brock400_4	33,00	400	59765

gen200_p0.9_44	44,00	200	17910
gen200_p0.9_55	55,00	200	17910
gen400_p0.9_55	55,00	400	71820
gen400_p0.9_65	65,00	400	71820
gen400_p0.9_75	75,00	400	71820
p_hat300-2	25,00	300	21928
p_hat300-3	36,00	300	33390
C250.9	44,00	250	27984
C125.9	34,00	125	6963

Tabel 3.3 DIMACS Dengan Graf Sedang

Dataset	***Max Clique***	**Simpul**	**Sisi**
DSJC500_5	13,00	500	112332
C500.9	57,00	500	112332
brock800_2	29,00	800	208166
brock800_4	33,00	800	207643
keller5	27,00	776	225990
p_hat700-1	11,00	700	60999
p_hat700-2	44,00	700	121728
p_hat700-3	62,00	700	183010

Tabel 3.4 DIMACS Dengan Graf Besar

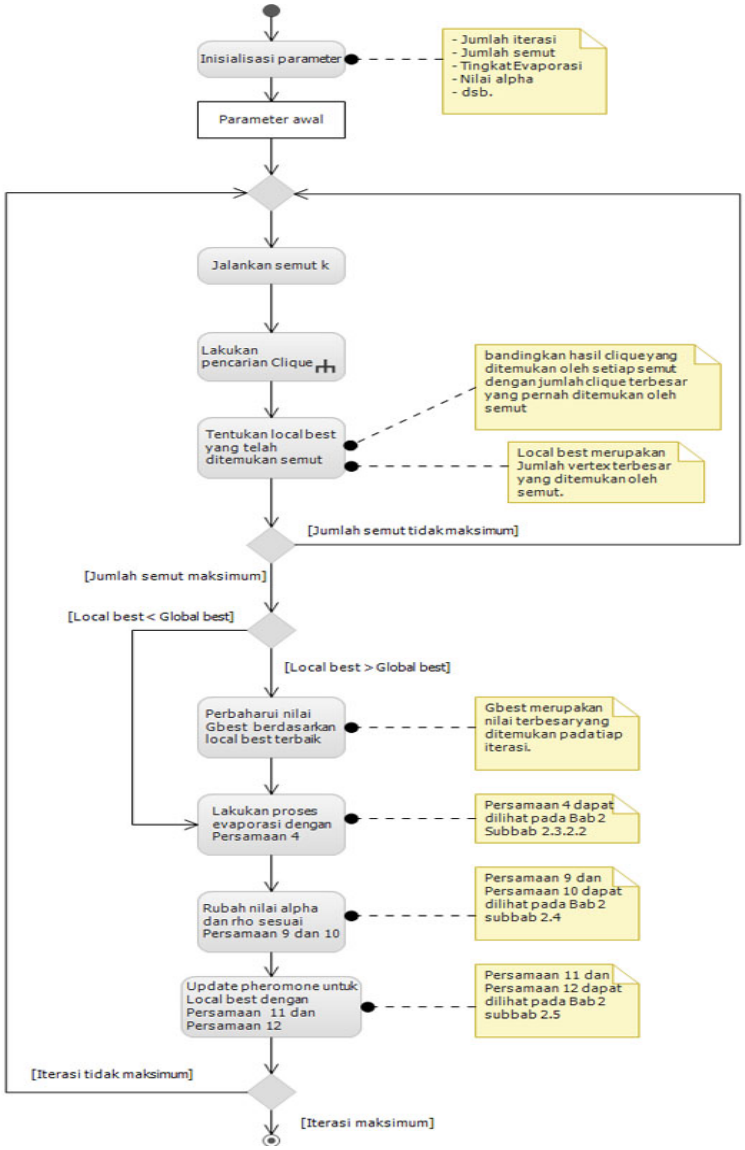
Dataset	***Max Clique***	**Simpul**	**Sisi**
MANN_a45	345,00	1035	533115
p_hat1500-2	65,00	1500	568960
p_hat1500-3	94,00	1500	847244
DSJC1000_5	15,00	1000	499652

3.3 Analisis

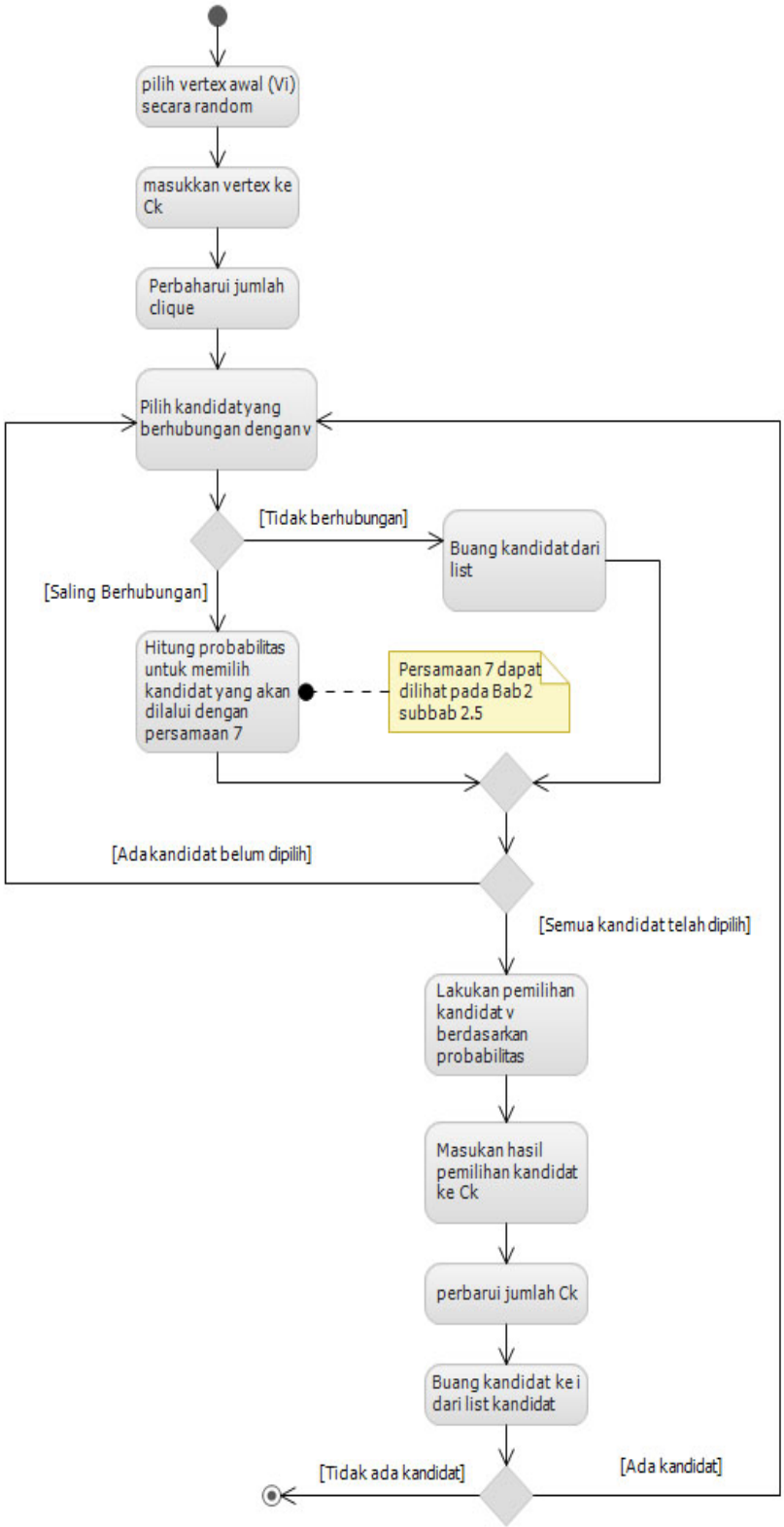
3.1.1 Analisis Proses

Tahap pertama dalam proses pencarian *clique* oleh algoritma *Improved Ant Colony Optimization* ditambah *Particle Swarm Optimization* adalah inisialisasi parameter awal. Parameter awal dapat mempengaruhi efektivitas algoritma *Improved Ant Colony Optimization* ditambah *Particle Swarm Optimization* dalam mencari *maximum clique*. Setelah parameter awal telah diinisialisasi, maka algortima akan mulai mencari *clique* yang ada pada graf.

Pencarian setiap semut akan dibandingkan untuk mendapatkan jumlah *clique* terbesar yang pernah ditemukan oleh semut (local best) dan kemudian dibandingkan dengan jumlah simpul terbanyak yang pernah ditemukan pada setiap iterasi (G_{best}). Proses algoritma *Improved Ant Colony Optimization* ditambah *Particle Swarm Optimization* dapat dilihat pada Gambar 3.2. sedangkan Gambar 3.3. merupakan *subactivit y* untuk mencari *clique* yang dilakukan oleh setiap semut.



Gambar 3.2 *Activity Diagram* Pencarian *Maximum Clique* dengan Algoritma *Improved Ant Colony Optimization* Ditambah *Particle Swarm Optimization*



Gambar 3.3 *Subactivity* Diagram *Maximum Clique* dalam Melakukan Pencarian *Clique*

Dalam mencari *maximum clique*, algoritma *Improved Ant Colony Optimization* ditambah *Particle Swarm Optimization* akan menggunakan Persamaan 7 untuk menentukan simpul kandidat yang akan dilalui, Persamaan 4 untuk proses evaporasi, dan Persamaan 10 untuk membarui *pheromone* yang telah ada.

Bentuk Persamaan 7 yang digunakan dalam penentuan kandidat yang akan dilalui

Keterangan:

= Kemungkinan perpindahan antar simpul.

= Jumlah *pheromone* yang ada pada sisi yang akan dilalui.

= Total *pheromone* yang ada pada kandidat.

= Variabel temperatur agar semut dapat mencapai posisi stabil

Degree = Jumlah sisi yang menghubungkan simpul kandidat.

EdgeNum = Jumlah sisi yang ada pada graf.

= Faktor pengurangan

Perhitungan probabilitas jalan yang akan dipilih oleh semut akan dilakukan secara *incremental* seperti yang telah dijelaskan pada bab 2 sebelumnya.

Bentuk Persamaan 10 dalam membarui *pheromone*.

Keterangan:

= Jumlah penambahan *pheromone* baru.

= Jumlah penambahan *pheromone* saat ini.

= Jumlah *pheromone* baru pada sisi antara titik i dan j.

= Jumlah *pheromone* saat ini pada sisi antara titik i dan j.

= Penambahan *pheromone*.

= Penambahan *pheromone* baru.

= Parameter pembelajaran.

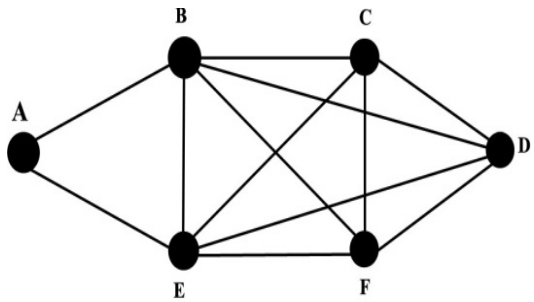
= Nilai acak antara 0 – 1.

= Jumlah simpul *clique* terbesar yang ditemukan oleh semut.

= Jumlah simpul *clique* terbesar yang ditemukan pada iterasi.

Proses perhitungan *Improved Ant Colony Optimization* ditambah *Particle Swarm Optimization*:

1. Misalkan graf yang akan dilakukan pencarian *maximum clique* ditunjukkan dengan Gambar 3.3



Gambar 3.4 Contoh Graf

a. Bentuk matriks dari Gambar 3.4

b. Matriks *pheromone* awal dianggap sama. Misalkan nilai awal = 1.

c. Jumlah semut = 3

d. = 1

e. = 0.05

f. = 0.95

g. = 1.0

h. V= 0

i. = 0

j. = 0.3

k. = 0.7

l.

m. *Local best* = 0

n. = 0.0002

2. Tentukan jumlah derajat untuk setiap simpul dengan menghitung jumlah angka satu pada matriks dan tentukan jumlah semua sisi yang ada pada graf dengan menjumlahkan semua derajat pada setiap simpul kemudian dibagi .

   A = 2

   B = 5

   C = 4

   D = 4

   E = 5

   F = 4

   Total semua simpul : = 12

3. Jalankan semut pertama.
4. Lakukan pencarian *clique* dengan menentukan simpul awal secara acak. Misalkan simpul awal yang terpilih adalah simpul A.
5. Masukan simpul A kedalam C1

   C1 = {A}

6. Pilih kandidat yang berhubungan dengan simpul A.

   Kandidat = {B,E}

7. Hitung probabilitas untuk dilalui oleh semut 1. Perhitungan awal dilakukan secara *incremental*. Kemudian hasil perhitungan *incremental* akan dibagi dengan total *pheromone* pada kandidat.

   Perhitungan awal

   P(B) = 1

P(E) = = 2

Pembagian dengan total *pheromone* kandidat

P(B) = ½ = 0.5 + = 0.917

P(E) = 2/2 = 1+ = 1.417

8. Hasilkan nilai acak r untuk menentukan simpul yang akan dipilih. Misalkan nilai r = 0.875 maka, bandingkan nilai r dengan probabilitas yang ada. Misalkan hasil pemilihan didapat nilai B.
9. Perbarui nilai

   = 0.95
10. Jalankan semut ke simpul B. Masukan simpul B ke C1.

    C1 = {A,B}
11. Keluarkan simpul B dari kandidat.

    Kandidat = {E}
12. Cek hubungan antar simpul B dengan simpul E. Jika berhubungan, hitung kemungkinan untuk simpul yang bersangkutan. Jika tidak, keluarkan simpul dari kandidat.

    {B,E} = 1 (Ya)
13. Hitung kemungkinan simpul dalam kandidat
14. Hitung kemungkinan dipilihnya setiap simpul dalam kandidat

    P(E)= = 1.396
15. Perbarui nilai

    = 0.903
16. Hasilkan nilai acak r. Misalkan r = 0.640. maka semut memilih simpul E.
17. Perbarui nilai

18. Masukan E kedalam C1.

    C1 = {A,B,E}
19. Keluarkan E dari kandidat

    Kandidat = { }
20. Kandidat kosong, pencarian semut pertama selesai. Bandingkan hasil yang ditemukan semut pertama dengan local best.

C1 berisi 3 simpul, sedangkan *local best* masih 0. Perbarui nilai *local best* dengan jumlah C1. Sehingga *local best* = 3.

21. Lakukan langkah yang sama untuk semut 2 dan 3.
22. Misalkan setiap semut telah melakukan penelusuran dan hasil *local best* yang didapat adalah 5 dan simpul yang termasuk *local best*

    Simpul *local best* = (C-D-F-E-B)
23. Bandingkan nilai *global best* dengan *local best*.

    (Local best > Global best) => (5 > 0) ya

    Perbarui nilai *global best* dengan *local best* sehingga nilai *global best* sekarang adalah 5.
24. Lakukan perbaruan *pheromone*. Misalkan nilai $r_1 = 0.255$ dan $r_2 = 0.871$
    a. Lakukan evaporasi untuk semua sisi

       = 0.05

    b. Perbaruan *pheromone* untuk setiap sisi yang menghubungkan simpul *local best* (C-D-F-E-B)

       = 0.03431

       Jumlah *pheromone* baru adalah

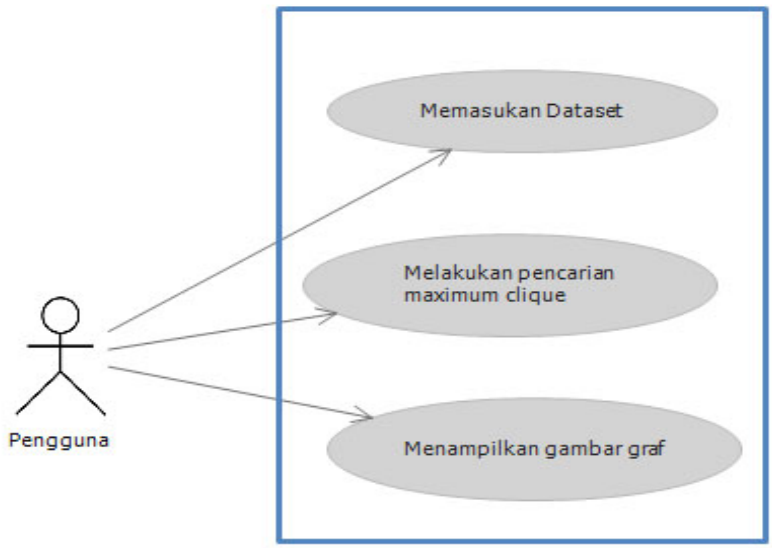
    c. Perbarui nilai alpha dengan mengecek jumlah iterasi sesuai dengan ketentuan pada Persamaan 9.
    d. Perbarui nilai rho

       = (1-0.0002)0.95 = 0.94981

       Nilai rho yang akan digunakan untuk evaporasi pada iterasi selajutnya adalah 0.94981.
25. Lakukan langkah awal sampai iterasi maksimum. Setelah iterasi maksimum, tampilkan *global best* yang telah didapat. Dalam kasus ini jumlah *global best* adalah 5.

3.3.1 Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional digambarkan dengan menggunakan *use case diagram*.



Gambar 3.5 *USE CASE* Diagram Pencarian *Maximum Clique* dengan Algoritma *Improved Ant Colony Optimization* Ditambah *Particle Swarm Optimization*

Skenario dari *use case* pada Gambar 3.5 akan ditunjukan pada Tabel 3.5, Tabel 3.6, dan Tabel 3.7.

Tabel 3.5 Skenario *USE CASE* Memasukan *Dataset*

Nama *Use Case*	Memasukan *dataset*	
Aktor	Pengguna	
Deskripsi	*Use case* ini untuk menginput graf yang akan dilakukan pencarian	
Prakondisi	Tidak ada graf yang terinput	
Pemicu *Use Case*	*Use case* akan bekerja setelah memilih tombol masukan graf.	
Event utama	Aksi Aktor	Respon sistem
	1. Pengguna memasukan jumlah simpul dari graf	3. Sistem memunculkan tabel sesuai dengan jumlah simpul.
	2. Pengguna memilih tombol bentuk tabel	6. Sistem *mengenerate* hubungan antar simpul
	4. Pengguna menginput hubungan antar simpul yang ditandai dengan 0	

	dan 1.	
	5. Pengguna menekan tombol *generate* matriks	
Event alternatif	Alt 4: Pengguna memasukan hubungan lebih dari 1 atau kurang dari 0. Sistem akan memberikan pemberitahuan dan meminta kembali pengguna untuk memasukan hubungan antar simpul.	
Kesimpulan	Graf yang digunakan adalah graf hasil penginputan pengguna	
Pascakondisi	Graf telah dimasukan	

Tabel 3.6 Skenario *USE CASE* Mencari *Maximum Clique*

Nama *Use Case*	Mencari *Maximum Clique*	
Aktor	Pengguna	
Deskripsi	*Use case* ini untuk mencari *maximum clique* dari graf	
Prakondisi	*Maximum clique* dari graf belum ditemukan	
Pemicu *Use Case*	*Use case* akan bekerja setelah pengguna selesai memilih *dataset*, menentukan parameter, dan menekan tombol mulai.	
Event utama	Aksi actor	Respon sistem
	1. Pengguna memilih *dataset* yang akan digunakan	4. Sistem melakukan penelusuran graf untuk mencari *maximum clique*
	5. Pengguna memasukan parameter awal	
	6. Pengguna menekan tombol mulai	
Event alternatif	Alt 1: Pengguna belum memilih *dataset* yang akan digunakan, maka sistem akan memunculkan pesan notifikasi untuk memasukan *dataset*.	
	Alt 2: Parameter awal yang dimasukan tidak sesuai. Sistem akan memberitahukan pengguna untuk mengisi kembali parameter awal.	
Kesimpulan	Sistem akan mulai mencari *maximum clique* saat *dataset* dan parameter awal telah ditetapkan	
Pascakondisi	Jumlah *maximum clique* ditemukan	

Tabel 3.7 Skenario *USE CASE* Menampilkan Gambar Graf

Nama *Use Case*	Menampilkan gambar graf
Aktor	Pengguna
Deskripsi	*Use case* ini untuk menampilkan gambar graf dari hasil masukan pengguna
Prakondisi	Gambar graf belum ditampilkan, *dataset* telah

	dimasukan	
Pemicu *Use Case*	*Use case* akan bekerja setelah pengguna menekan tombol tampilkan gambar dan *dataset* telah dimasukan.	
Event utama	Aksi aktor	Respon sistem
	1. Pengguna menekan tombol tampilkan graf	2. Sistem menampilkan graf
Event alternatif	Alt 1: Pengguna belum memasukan *dataset*, maka sistem akan memberikan notifikasi untuk memasukan *dataset* terlebih dahulu.	
Kesimpulan	Sistem akan menampilkan gambar dari graf jika pengguna telah memasukan *dataset*.	
Pascakondisi	Gambar graf dari *dataset* ditampikan	

3.3.2 Analisis Kebutuhan Non-Fungsional

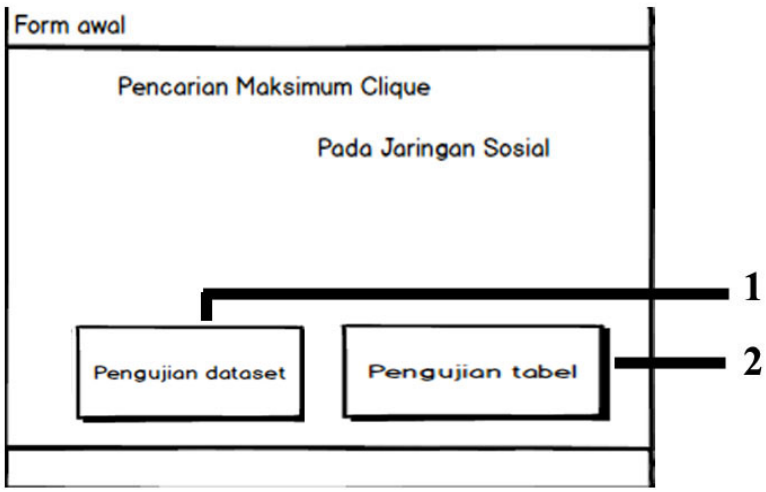
Persyaratan analisis Non-fungsional dari sistem pencarian *maximum clique* yang dilakukan dengan algoritma *Improved Ant Colony Optimization* ditambah *Particle Swarm Optimization* adalah sebagai berikut:

a. Sistem dapat memberikan pemberitahuan jika proses yang dikerjakan sudah selesai.
b. Sistem dapat memberikan informasi maksimum clique yang ada pada *dataset* berupa gambar.
c. Aplikasi akan menampilkan peringatan ketika terjadi kesalahan pada pemasukan data.
d. Aplikasi akan menampilkan peringatan ketika *dataset* yang dimasukan tidak sesuai dengan format yang telah ditentukan.

3.4 Perancangan

3.4.1 Perancangan Tampilan

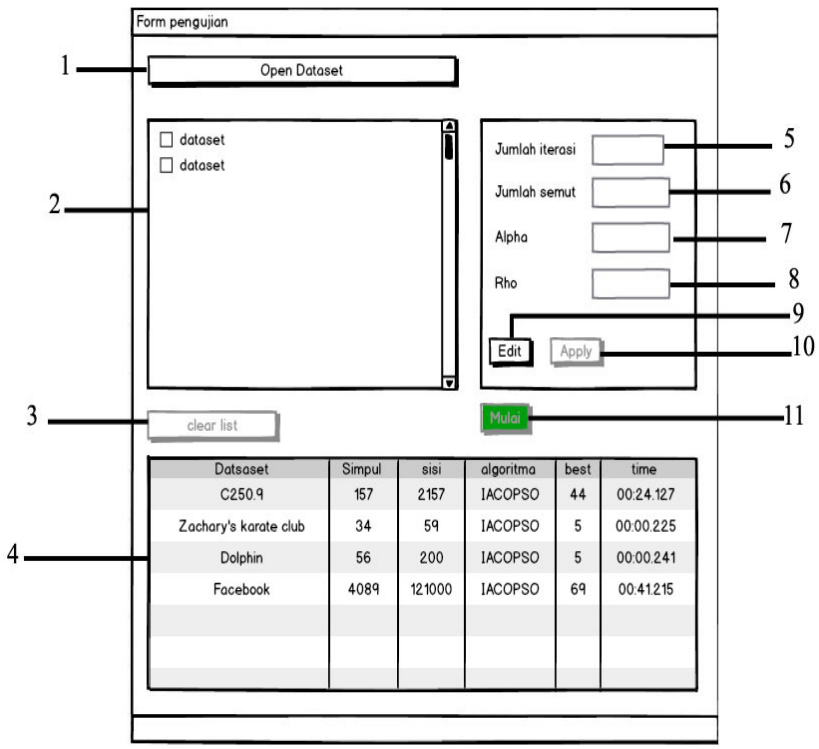
Perancangan pada tugas akhir ini adalah sebagai berikut:



Gambar 3.6 Tampilan *Form* Pencarian *Maximum Clique*

Keterangan Gambar 3.6:

1. Tombol Pengujian *Dataset* : Menampilkan *form* pengujian *dataset*
2. Tombol Pengujian Tabel : Menampilkan *form* pengujian tabel

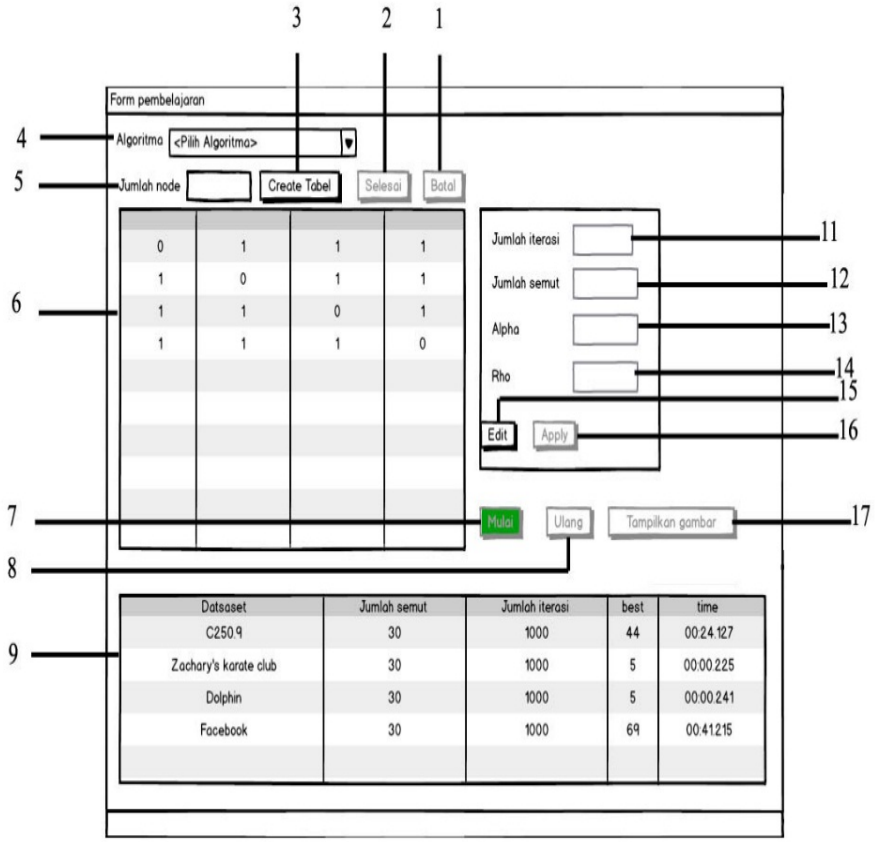


Gambar 3.7 Tampilan *Form* Pengujian *Dataset*

Keterangan Gambar 3.7:

1. *Open Dataset* : Memulih *dataset* yang akan diuji

2. *Checked Listbox* : Berisi *dataset* yang akan diuji
3. *Clear List* : Menghapus isi dari *checked listbox*
4. *Datagridview* : Menampilkan hasil dari pengujian
5. Jumlah Iterasi : Menentukan jumlah iterasi
6. Jumlah Semut : Menentukan jumlah semut
7. Alpha : Menentukan nilai alpha
8. Rho : Menentukan nilai rho
9. *Edit* : Merubah nilai dari iterasi, semut, alpha, dan rho
10. *Apply* : Menetapkan perubahan nilai dari iterasi, semut, alpha, dan rho.
11. Mulai : Menjalankan pencarian *maximum clique*.

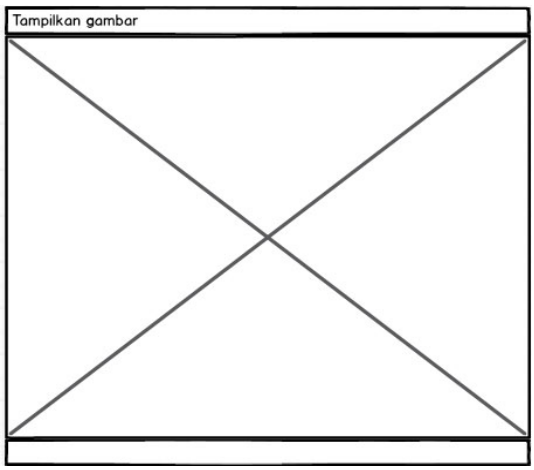


Gambar 3.8 Tampilan *Form* Pengujian Tabel

Keterangan Gambar 3.8:

1. Batal : Membatalkan hasil masukan pengguna

2. Selesai : Menetapkan hasil masukan pengguna
3. *Create Table* : Menampilkan *datagridview* sesuai dengan jumlah simpul
4. Algoritma : Memilih jenis algoritma
5. Jumlah *node* : menentukan jumlah simpul dari graf yang akan diuji
6. *Datagridview hub* : memasukan hubungan antar graf
7. Mulai : menjalankan pencarian
8. Ulang : memulai kembali pengimputan dari awal
9. *Datagridview* hasil : Menampilkan hasil dari pencarian
10. Proses Perhitungan : Tempat menampilkan proses perhitungan
11. Jumlah iterasi : Menetapkan jumlah iterasi.
12. Jumlah semut : Menentapkan jumlah semut
13. Alpha : Menetapkan nilai Alpha
14. Rho : Menetapkan nilai rho
15. *Edit* : Merubah nilai dari jumlah iterasi, semut, alpha, dan rho.
16. *Apply* : Menetapkan hasil perubahan
17. Tampilkan gambar : Menampilkan *form* gambar



Gambar 3.9 Tampilan *Form* Gambar

Form pada gambar 3.9 bertujuan untuk menampilkan gambar dari *dataset* berupa graf.

3.5 Pengkodean

Pengkodean akan dilakukan dengan bahasa *C#* dengan menggunakan plat*form windows form*.

3.6 Pengujian

Proses pengujian akan dilakukan sebagai berikut:

a. Pengujian awal dilakukan dengan mencari nilai rho (tingkat evaporasi) untuk algoritma *Improved Ant Colony Optimization* ditambah *Particle Swarm Optimization* (IACOPSO) dimana nilai yang diuji terdiri dari 0.1, 0.3, 0.5, 0.7, 0.9 dan 1 ( ). *Dataset* yang digunakan adalah *dataset* jaringan sosial dan *dataset* DIMACS. Untuk mempermudah pengujian dan analisis hasil maka graf DIMACS akan dibagi menjadi 3 kelompok yaitu graf kecil, graf sedang, dan graf besar. Tabel dari kelompok *dataset* DIMACS dapat dilihat pada bab 3 subbab 3.2 Pengumpulan data.
b. Setelah menemukan nilai rho yang sesuai, maka parameter rho akan ditetapkan untuk algoritma *Improved Ant Colony Optimization* ditambah *Particle Swarm Optimization* (IACOPSO) dalam mencari maksimum *clique* pada *dataset* jaringan sosial dan DIMACS.
c. Untuk mengetahui peformansi dari algoritma IACOPSO, maka hasil dari algoritma IACOPSO akan dibandingkan dengan hasil dari algoritma ACO, *improved* ACO (IACO), dan ACO gabung PSO (ACOPSO) dengan menggunakan *dataset* yang sama.

3.7 Menarik Kesimpulan

Kesimpulan akan diambil berdasarkan hasil dari pengujian yang telah dilakukan.

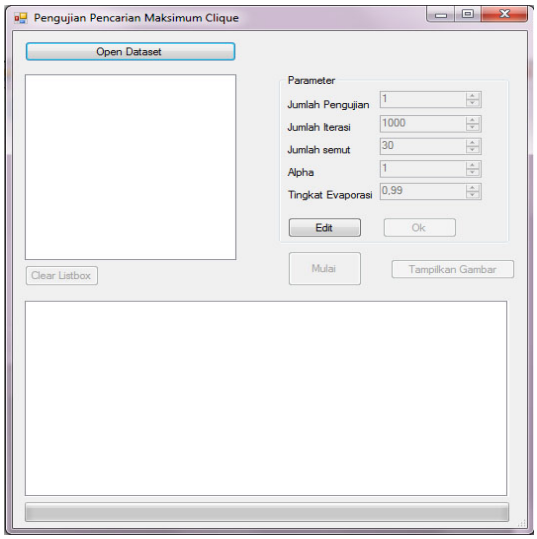
BAB IV
HASIL DAN PEMBAHASAN

4.1 Hasil

Pada aplikasi TA ini terdapat dua mode, yaitu mode pengujian *dataset* dan mode pengujian dengan tabel.

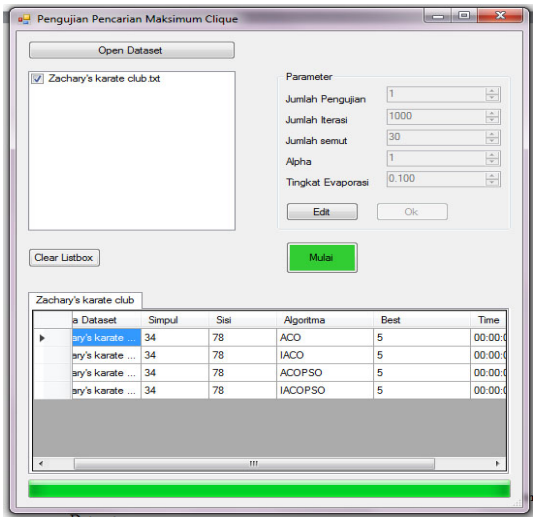
Gambar 4.1 Tampilan Awal Program

Pada gambar 4.1, pengguna dapat memilih mode pengujian dengan *dataset* ataupun mode pengujian dengan tabel.



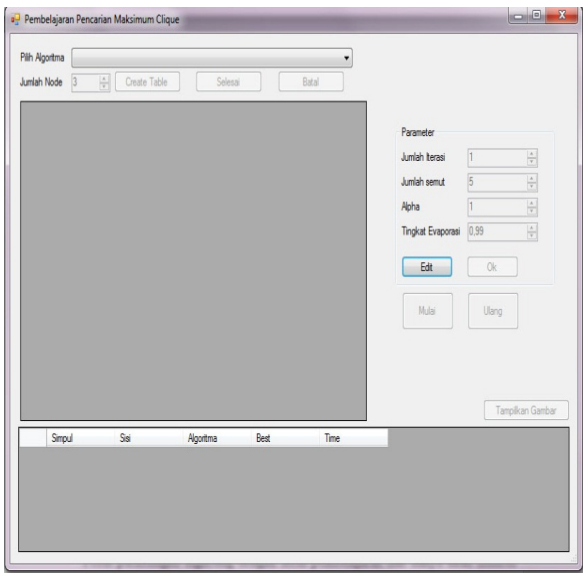
Gambar 4.2 Tampilan *Form* Pengujian *Dataset*

Form pengujian *dataset* digunakan untuk melakukan pengujian dengan *dataset* dimana *dataset* dapat berupa ekstensi .txt dan .clq



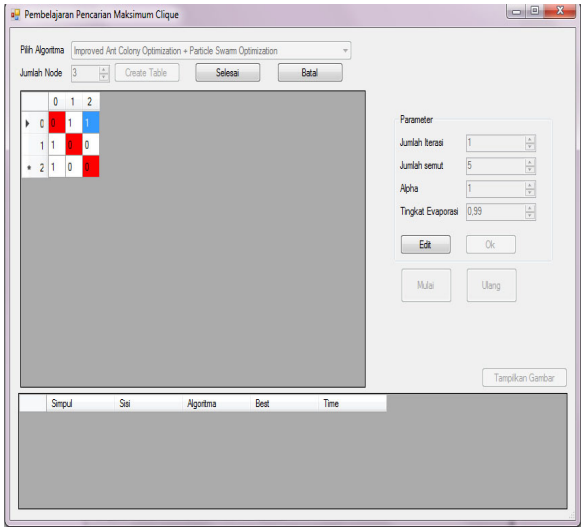
Gambar 4.3 Tampilan Hasil *Form* Pengujian *Dataset*

Gambar 4.3 memperlihatkan hasil yang ditampilkan oleh *Form* Pengujian *Dataset*. Tampilan hasil terdiri dari nama *dataset* yang diuji, jumlah simpul, jumlah sisi, algoritma, maksimum *clique* yang ditemukan, dan waktu yang dibutuhkan untuk menemukan maksimum *clique*.



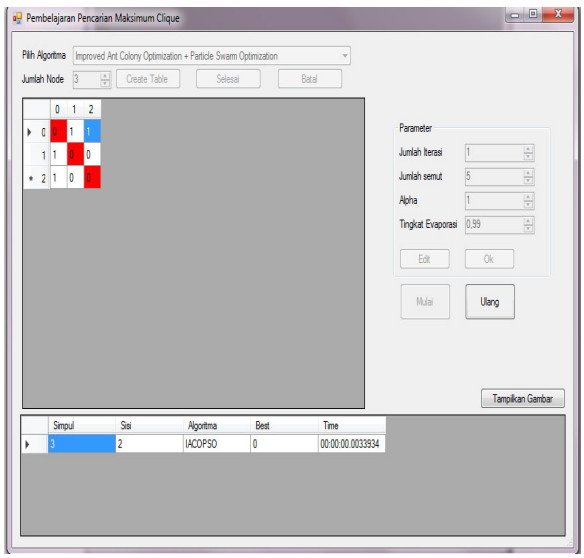
Gambar 4.4 Tampilan *Form* Pengujian Tabel

Form pengujian tabel digunakan untuk melakukan pencarian dari hasil masukan pengguna yang berupa tabel.



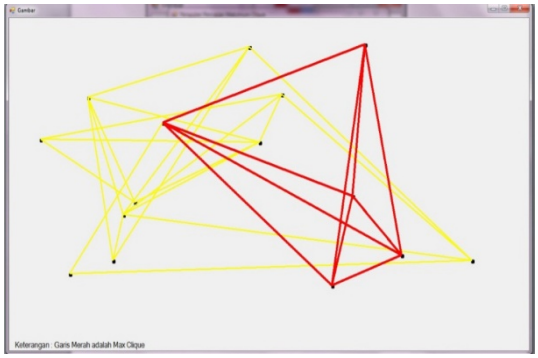
Gambar 4.5 Tampilan *Form* Pengujian Tabel ketika Memasukkan Data

Gambar 4.5 memperlihatkan data dimasukkan di dalam tabel dimana nilai 0 menandakan tidak ada hubungan antar simpul dan nilai 1 menandakan terdapat hubungan antar simpul.



Gambar 4.6 Tampilan Hasil Keluaran *Form* Pengujian Tabel

Setelah pencarian maksimum *clique* selesai maka hasil keluarannya dapat dilihat pada Gambar 4.6.



Gambar 4.7 Tampilan *Form* Gambar

Setelah hasil keluaran pada *Form* Tabel ditampilkan, pengguna dapat menekan tombol tampilkan gambar. *Form* Tampilan gambar dapat dilihat pada Gambar 4.7

4.2 Pembahasan

Spesifikasi perangkat keras yang digunakan dalam pengujian adalah sebagai berikut:

a. AMD FX(tm) -8350 Eight-Core Processor 4.00Ghz
b. RAM 16.0 GB

Sedangkan spesifikasi perangkat lunak yang digunakan adalah:

a. Windows 7 Ultimate 64-bit
b. Microsoft Visual Studio 2010

Dataset yang akan diuji dapat dilihat pada Tabel 4.1

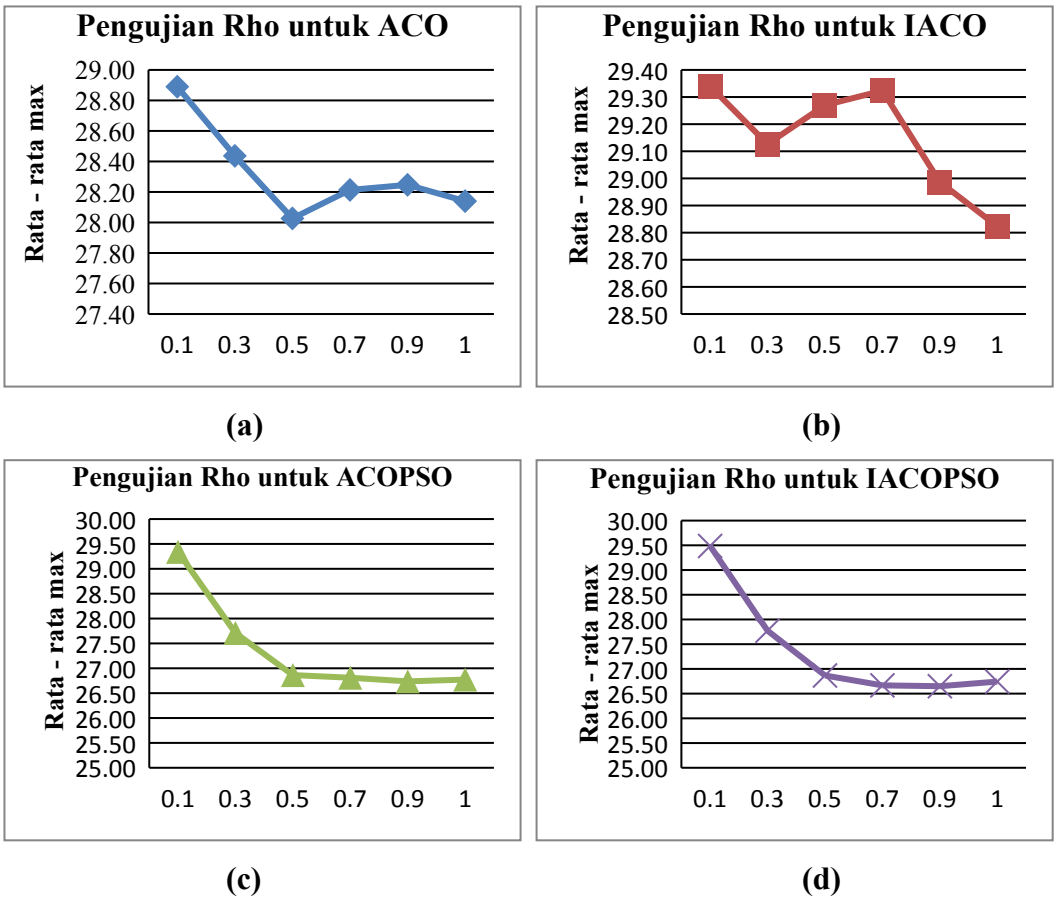
Tabel 4.1 *Dataset* yang Diuji

Kategori *Dataset*	Nama *Dataset*	*Max Clique*	Simpul	Sisi
Social Network	*Zachary's karate club*	-	34	78
	Social network of dolphins	-	62	159
	Facebook Social circle	-	4.039	88.234
	Erdos Collaboration Network 991	-	492	1417
	Erdos Collaboration Network 971	-	472	1314
	Co-authorship of scientist in network theory and experiment	-	1589	1190
	Email Interchange Network,University of Rovia I Virgili, Tarragona	-	1133	5451

Graf Kecil	*p_hat300-1*	8	300	10 933
	brock200_2	12	200	9 876
	hamming8-4	16	256	20 864
	brock200_4	17	200	13 089
	brock400_2	29	400	59 786
	brock400_4	33	400	59 765
	gen200_p0.9_44	44	200	17 910
	gen200_p0.9_55	55	200	17 910
	gen400_p0.9_55	55	400	71 820
	gen400_p0.9_65	65	400	71 820
	gen400_p0.9_75	75	400	71 820
	p_hat300-2	25	300	21 928
	p_hat300-3	36	300	33 390
	C250.9	44*	250	27 984
	C125.9	34*	125	6 963
Graf Sedang	*brock800_2*	24	800	208 166
	brock800_4	26	800	207 643
	keller5	27	776	225 990
	p_hat700-1	11	700	60 999
	p_hat700-2	44	700	121 728
	p_hat700-3	62*	700	183 010
	DSJC500_5	13	500	125248
Graf Besar	*MANN_a45*	345	1 035	533 115
	hamming10-4	40	1 024	434 176
	p_hat1500-2	65*	1 500	568 960
	p_hat1500-3	94*	1 500	847 244

4.2.1 Pengujian Tingkat Evaporasi

Pengujian awal dilakukan dengan menentukan tingkat evaporasi yang akan digunakan. Tingkat evaporasi yang akan diuji terdiri dari 0.1, 0.3, 0.5, 0.7, 0.9, dan 1 ( ). Pengujian dilakukan sebanyak 10 dengan jumlah iterasi 1000, jumlah semut 30, dan alpha 1. Hasil pengujian dapat dilihat pada Gambar 4.8(a) untuk algoritma ACO, 4.8(b) untuk algoritma IACO, 4.8(c) untuk algoritma ACOPSO, dan 4.8(d) untuk algoritma IACOPSO.



Gambar 4.8 Pengujian Rho Masing-masing Algoritma

Berdasarkan Gambar 4.8(a), 4.8(b), 4.8(c), dan 4.8(d), tingkat evaporasi yang semakin rendah dapat menemukan maksimum *clique* yang lebih besar.

4.2.2 Hasil Pengujian pada *Dataset*

Pengujian akan dilakukan dengan menggunakan parameter sebagai berikut:

a. Alpha awal 1 dengan peningkatan sesuai Persamaan 9.

b. Rho 0.1 dengan perubahan nilai sesuai Persamaan 10

Nilai rho ditentukan bedasarkan hasil pengujian yang telah dilakukan pada *subbab* sebelumnya.

c.

d. Iterasi 1000
e. Jumlah semut 30
f. = 6
g. = 0.01
h. = 0.05
i. = 1.0
j. = 0.0002
k. = 0
l. = 0
m. = 0.3
n. = 1-

Hasil pengujian dapat dilihat pada Tabel 4.2, 4.3, 4.4 dan 4.5.

Tabel 4.2 Hasil Pengujian IACOPSO pada *Dataset* Jaringan Sosial

Hasil	*Best Known*	IACOPSO				
		Best	***Min***	***Stddev***	***Avgbest***	***Avgtime***
Co-authorship of scientist in network theory and experiment	20,00	20	20	0	20,00	00:00,621
Dolphin network	5,00	5	5	0	5,00	00:00,059
Email interchange network univ of rovia i virgili tarragon	12,00	12	12	0	12,00	00:00,870
Erdos991	7,00	7	7	0	7,00	00:00,219
Facebook_combined	69,00	69	69	0	69,00	00:12,341
Pajek network Erdos collaboration network 971	7,00	7	7	0	7,00	00:00,204
Zachary's karate club	5,00	5	5	0	5,00	00:00,047

Pada Tabel 4.2 IACOPSO dapat menemukan maksimum *clique* pada jaringan sosial dengan standard deviasi 0 dimana pengujian yang dilakukan sebanyak 10 kali memberikan hasil yang sama.

Tabel 4.3 Hasil Pengujian IACOPSO pada *Dataset* DIMACS Graf Kecil

Hasil	*Best Known*	IACOPSO				
		Best	***Min***	***Stddev***	***Avgbest***	***Avgtime***
p_hat300-1	8,00	8	8	0	8,00	00:01,432

brock200_2	12,00	12	12	0	12,00	00:01,635
hamming8-4	16,00	16	16	0	16,00	00:03,429
brock200_4	17,00	17	15	0,67	16,30	00:02,599
brock400_2	29,00	23	22	0,52	22,40	00:09,375
brock400_4	33,00	25	22	0,97	22,60	00:09,484
gen200_p0.9_44	44,00	44	39	1,89	43,00	00:05,763
gen200_p0.9_55	55,00	55	55	0	55,00	00:06,042
gen400_p0.9_55	55,00	52	49	0,97	50,40	00:15,457
gen400_p0.9_65	65,00	65	59	1,90	64,40	00:15,475
gen400_p0.9_75	75,00	75	75	0	75,00	00:16,087
p_hat300-2	25,00	25	25	0	25,00	00:04,119
p_hat300-3	36,00	36	34	0,97	35,40	00:06,759
C250.9	44,00	44	42	0,70	43,60	00:08,082
C125.9	34,00	34	34	0	34,00	00:03,164

Pada Tabel 4.3 IACOPSO mampu menemukan maksimum *clique* pada beberapa *dataset* dengan standar deviasi 0 akan tetapi terdapat beberapa *dataset* yang masih belum mendapatkan hasil maksimum.

Tabel 4.4 Hasil Pengujian IACOPSO pada *Dataset* DIMACS Graf Sedang

Hasil	***Best Known***	**IACOPSO**				
		Best	***Min***	***Stddev***	***Avgbest***	***Avgtime***
DSJC500_5	13,00	13	12	0,52	12,60	00:08,034
C500.9	57,00	55	47	3,03	51,50	00:22,668
brock800_2	29,00	19	18	0,52	18,60	00:27,360
brock800_4	33,00	20	18	0,67	18,70	00:26,305
keller5	27,00	27	24	0,88	25,10	00:30,076
p_hat700-1	11,00	10	9	0,48	9,70	00:07,138
p_hat700-2	44,00	44	44	0	44,00	00:16,382
p_hat700-3	62,00	62	61	0,32	61,90	00:29,178

Pada Tabel 4.4 IACOPSO mampu menemukan maksimum *clique* setara dengan maksimum *clique* yang pernah ditemukan pada sebagian *dataset* sedang

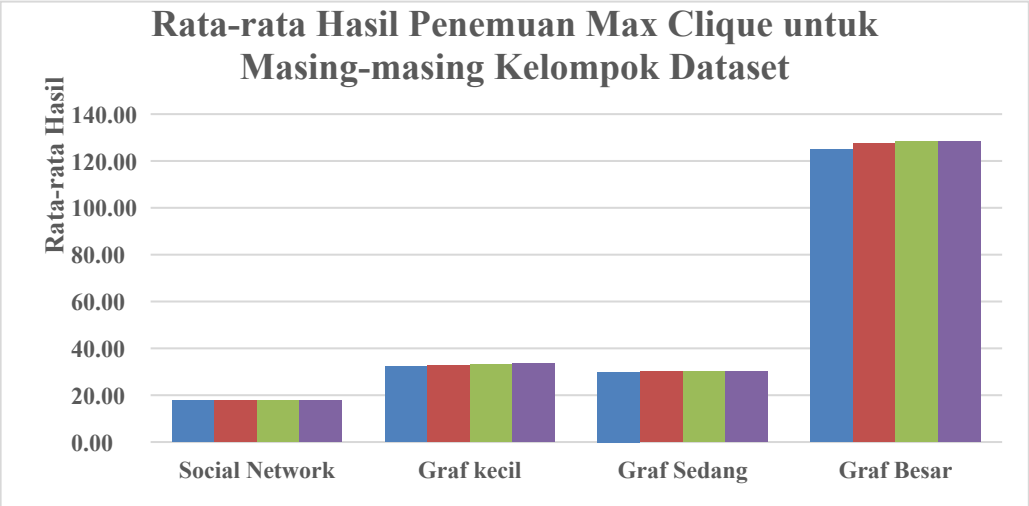
dengan standar deviasi yang tidak mencapai angka 0. Sedangkan dari segi waktu, IACOPSO membutuhkan waktu lebih lama dalam mencari maksimum *clique* pada graf sedang jika dibandingkan dengan graf kecil.

Tabel 4.5 Hasil Pengujian IACOPSO pada *Dataset* DIMACS Graf Besar

Hasil	*Best Known*	IACOPSO				
		Best	*Min*	*Stddev*	*Avgbest*	*Avgtime*
MANN_a45	345,00	344	341	0,95	341,70	06:26,058
p_hat1500-2	65,00	65	65	0	65,00	01:12,950
p_hat1500-3	94,00	94	93	0,48	93,30	02:01,322
DSJC1000_5	15,00	14	13	0,53	13,50	00:31,960

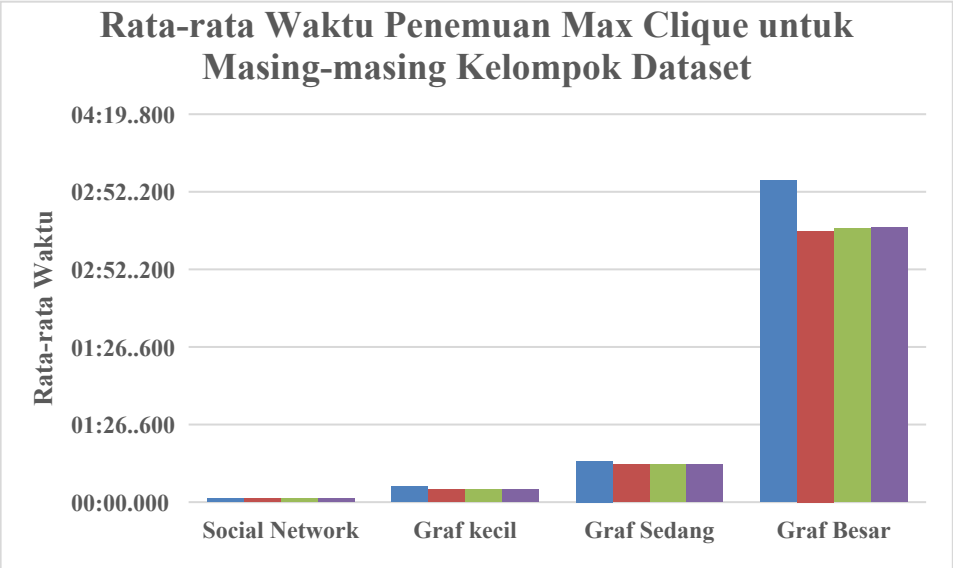
Pada Tabel 4.5 IACOPSO mampu menemukan maksimum *clique* setara dengan maksimum *clique* yang pernah ditemukan pada sebagian *dataset* besar dengan waktu yang lebih lama jika dibandingkan dengan mencari maksimum *clique* pada graf kecil dan sedang.

Untuk mengetahui kemampuan IACOPSO dalam mencari maksimum *clique*, maka hasil pencarian maksimum *clique* dan waktu yang dibutuhkan untuk mencari maksimum *clique* pada *dataset* akan dibandingkan dengan hasil yang didapat oleh algoritma lain.



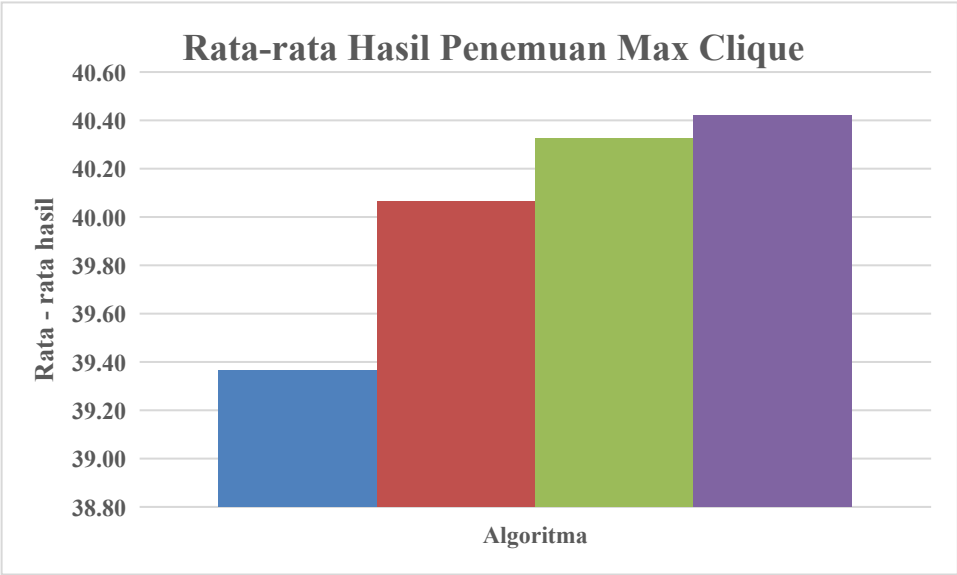
Gambar 4.9 Rata-rata Hasil Penemuan *Maximum Clique* pada Setiap Kelompok *Dataset*

Pada Gambar 4.9, dapat dilihat IACO menemukan rata-rata maksimum *clique* yang lebih tinggi dibandingkan dengan algoritma lain pada *dataset* graf sedang. Sedangkan IACOPSO menemukan rata-rata maksimum *clique* yang lebih tinggi pada graf kecil dan besar.



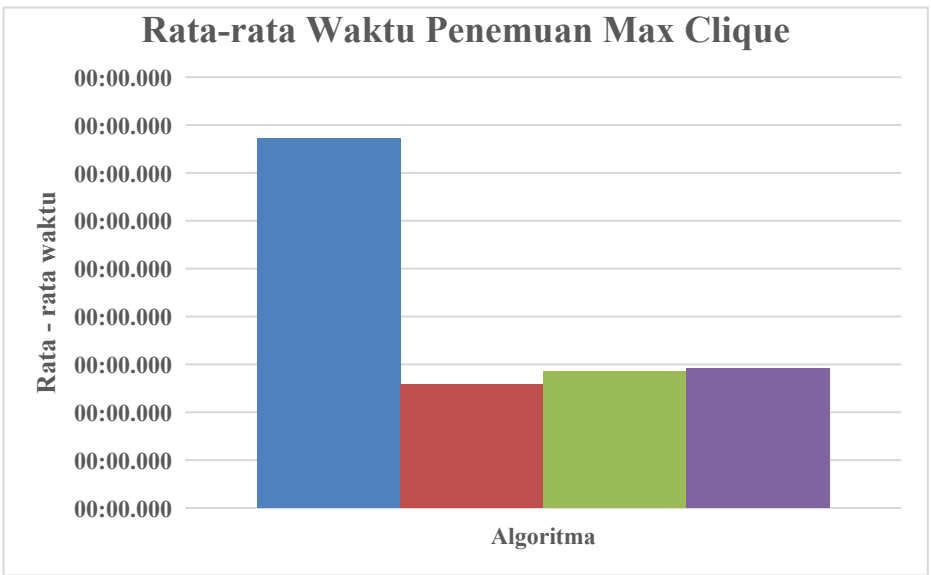
Gambar 4.10 Rata-rata Waktu Penemuan *Maximum Clique* pada Setiap Kelompok *Dataset*

Seperti yang dapat dilihat pada Gambar 4.10, waktu yang dibutuhkan oleh IACOPSO dalam mencari maksimum *clique* pada graf jaringan sosial, graf kecil, dan graf sedang lebih cepat dibandingkan dengan ACOPSO namun, lebih lambat jika dibandingkan dengan IACO. Untuk graf besar, IACOPSO lebih lambat jika dibandingkan dengan IACO dan ACOPSO.



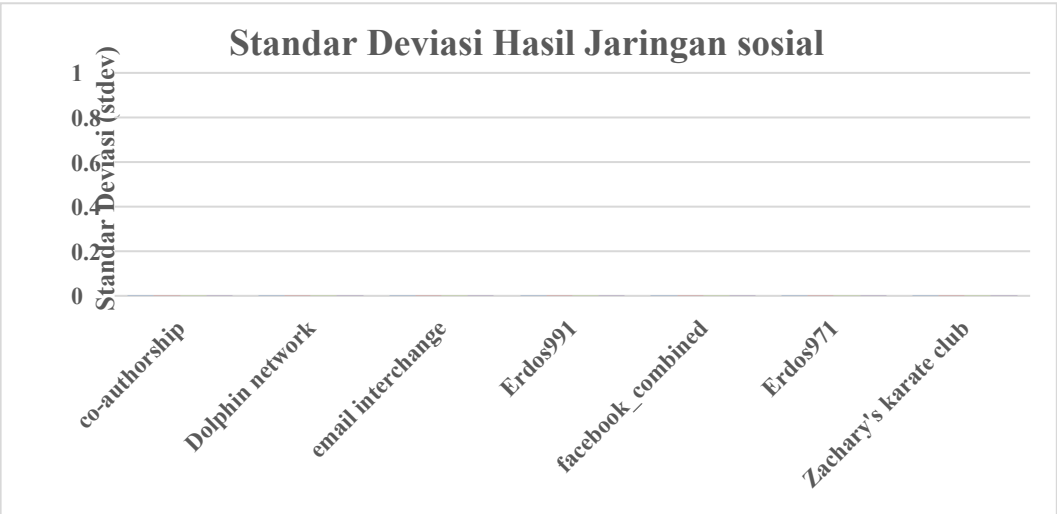
Gambar 4.11 Rata-rata Hasil Pencarian *Maximum Clique* dengan 4 Algoritma

Gambar 4.11 merupakan hasil rata-rata dari maksimum *clique* yang ditemukan pada semua *dataset*. Dari Gambar 4.11 dapat dilihat IACOPSO menemukan maksimum *clique* lebih besar jika dibandingkan dengan algoritma lain sedangkan dari sisi waktu dapat dilihat pada Gambar 4.12.



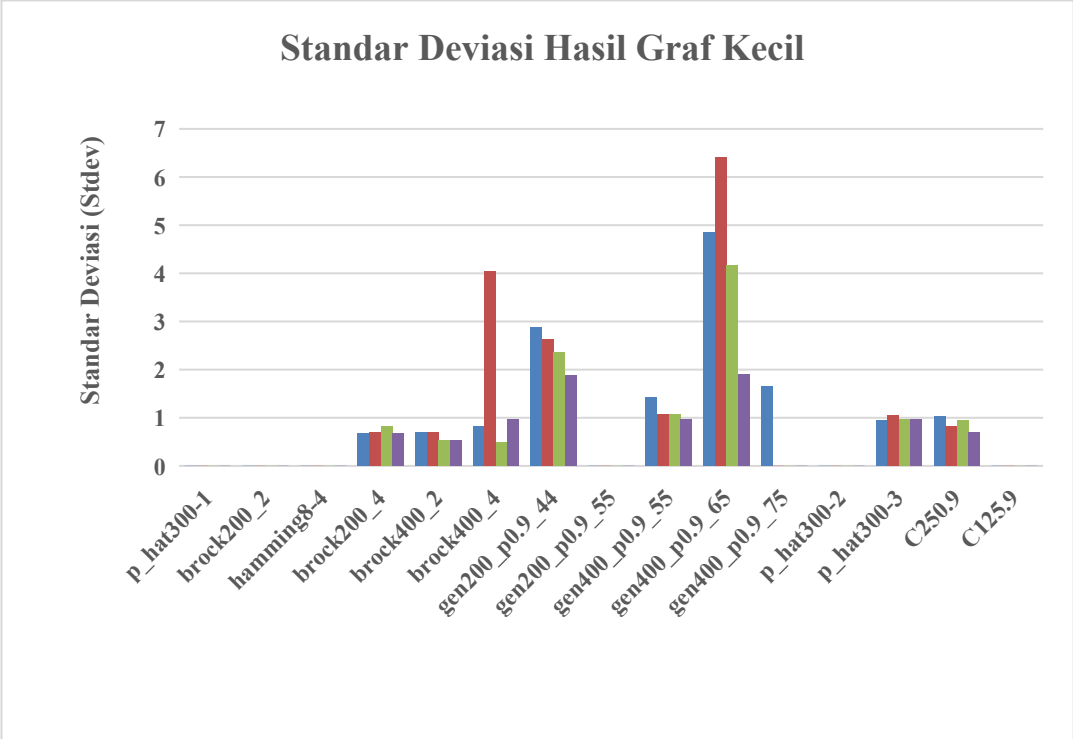
Gambar 4.12 Rata-rata Waktu Pencarian *Maximum Clique* dengan 4 Algoritma

Gambar 4.12 merupakan rata-rata waktu yang dibutuhkan untuk mencari maksimum *clique* pada semua *dataset* yang diuji. Dari Gambar 4.12 dapat dilihat IACOPSO membutuhkan waktu lebih lama dibandingkan dengan algoritma ACOPSO dan IACO akan tetapi lebih baik jika dibandingkan dengan ACO.



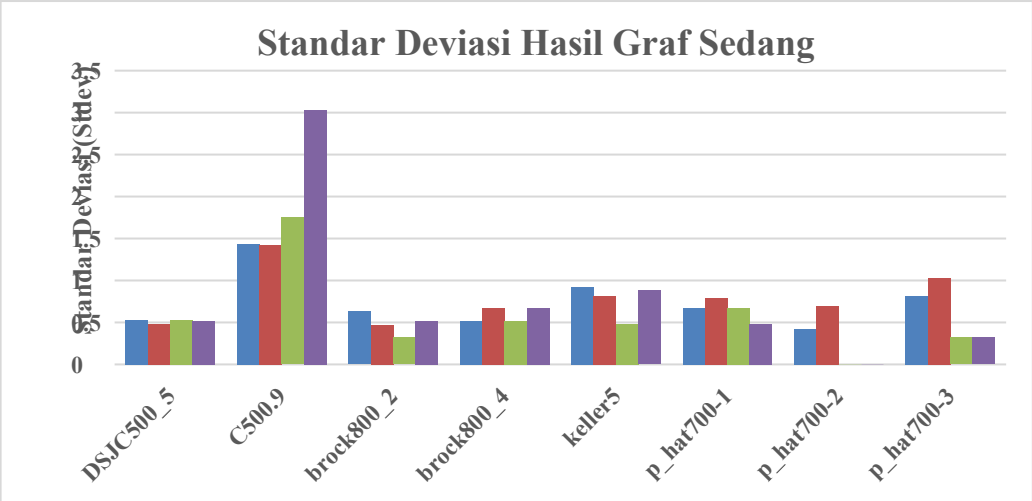
Gambar 4.13 Standar Deviasi Hasil pada Graf Jaringan Sosial

Dari Gambar 4.13, empat algoritma memiliki standar deviasi yang sama pada jaringan sosial yaitu 0.



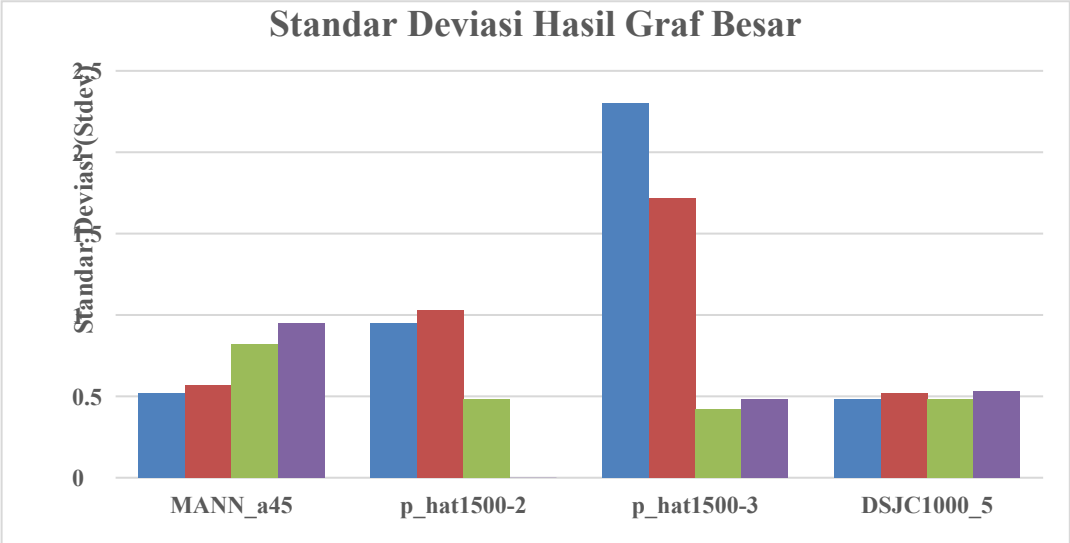
Gambar 4.14 Standar Deviasi Hasil pada Graf DIMACS Kecil

Dari Gambar 4.14, keempat algoritma memiliki standar deviasi 0 pada 4 *dataset*, namun standar deviasi IACOPSO lebih rendah dibandingkan dengan algoritma lain pada *dataset* gen200_p0.9_44, gen400_p0.9_55, gen400_p0.9_65, dan C250.9.



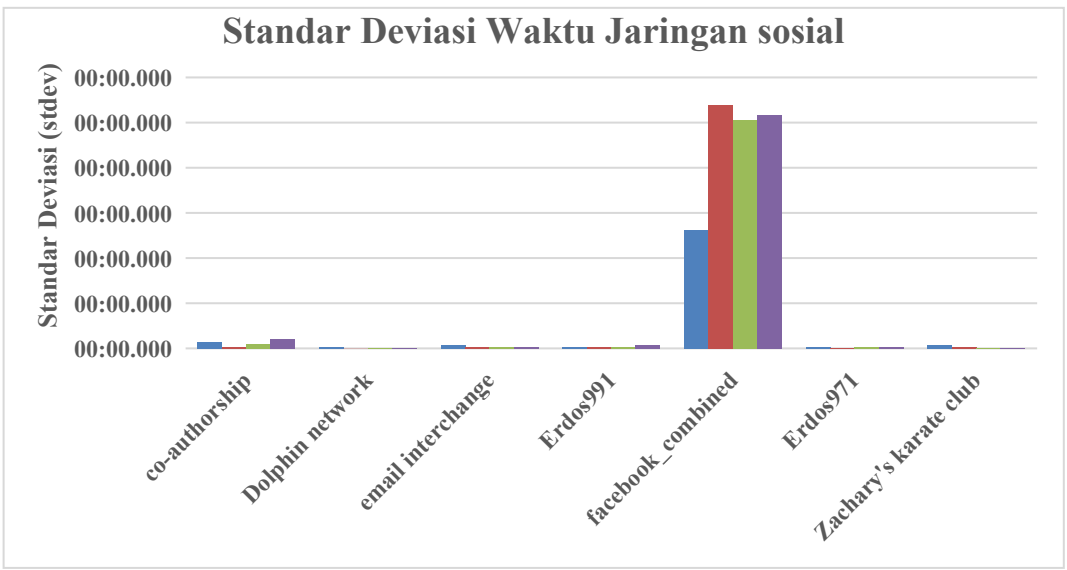
Gambar 4.15 Standar Deviasi Hasil pada Graf DIMACS Sedang

Dari Gambar 4.15, standar deviasi dari IACOPSO mencapai angka yang tinggi pada *dataset* C500.9, namun memiliki standar deviasi yang relatif lebih rendah pada beberapa *dataset* lain.



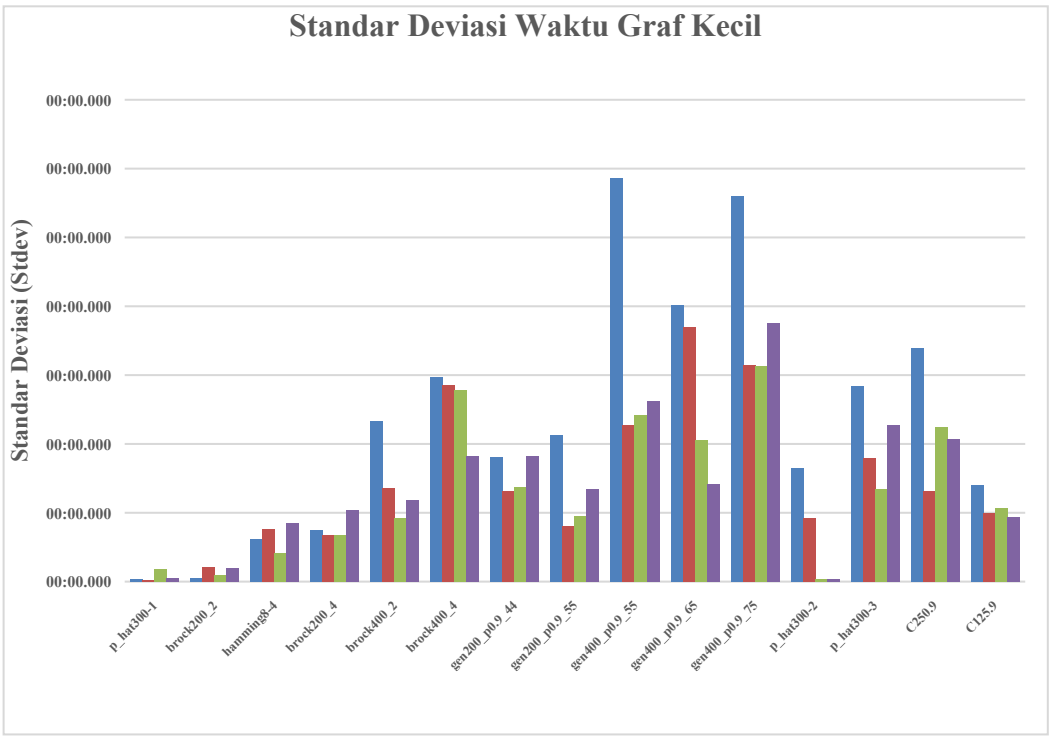
Gambar 4.16 Standar Deviasi Hasil pada Graf DIMACS Besar

Pada Gambar 4.16, IACOPSO memiliki standar deviasi yang lebih tinggi jika dibandingkan dengan algoritma ACOPSO, tetapi standar deviasi IACOPSO mencapai angka 0 pada *dataset* p_hat1500-2.



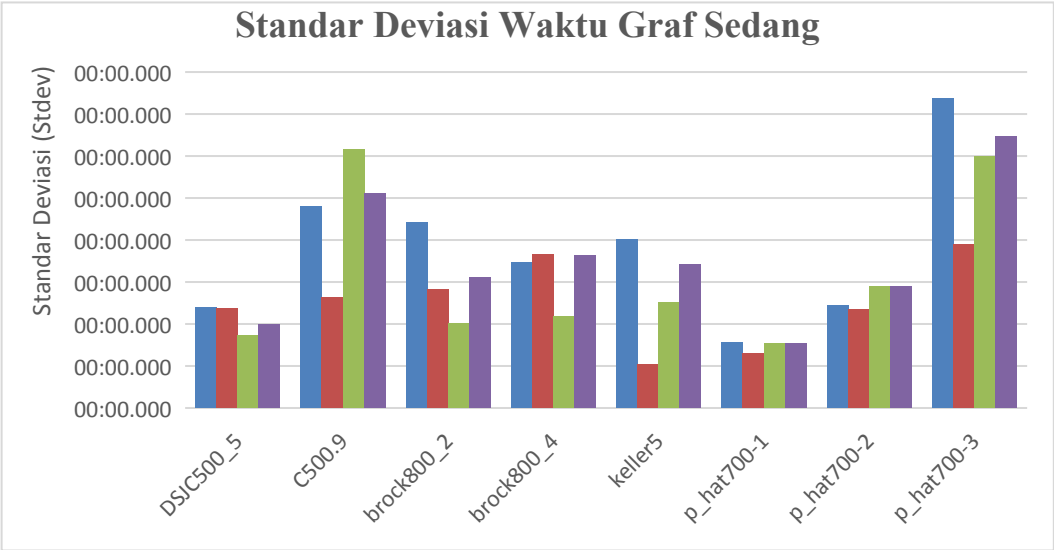
Gambar 4.17 Standar Deviasi Waktu pada Graf Jaringan Sosial

Pada Gambar 4.17, IACO memiliki standar deviasi waktu yang relatif rendah dibandingkan dengan algoritma lain.



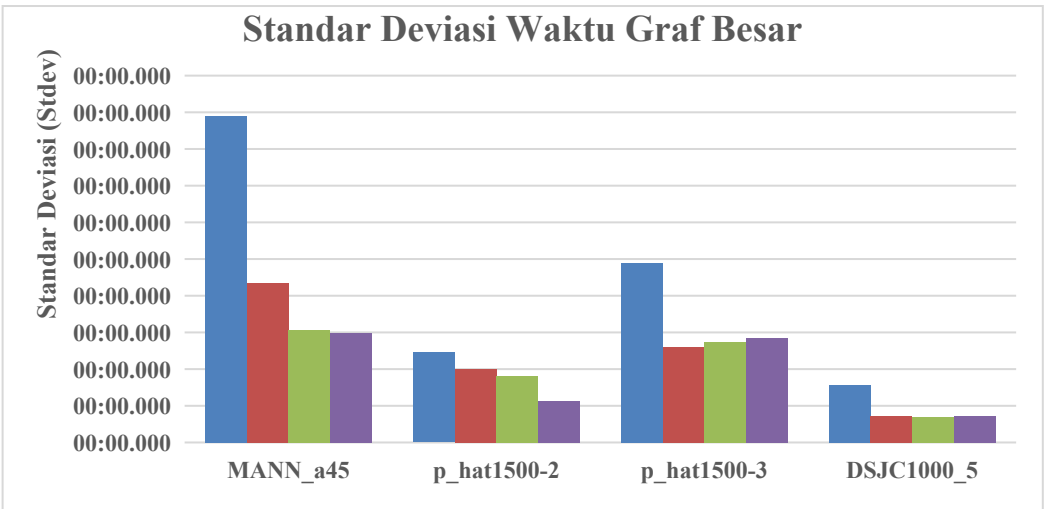
Gambar 4.18 Standar Deviasi Waktu pada Graf DIMACS Kecil

Pada Gambar 4.18, standar deviasi IACOPSO lebih rendah dibandingkan dengan ACO kecuali pada 4 *dataset* awal.



Gambar 4.19 Standar Deviasi Waktu pada Graf DIMACS Sedang

Pada Gambar 4.19, standar deviasi IACOPSO lebih tinggi jika dibandingkan dengan algoritma IACO dan ACOPSO, namun relatif lebih rendah jika dibandingkan dengan ACO.



Gambar 4.20 Standar Deviasi Waktu pada Graf DIMACS Besar

Pada Gambar 4.20, IACOPSO memiliki standar deviasi yang rendah dibandingkan dengan algoritma lain pada 2 *dataset* awal akan tetapi, lebih tinggi dibandingkan dengan IACO dan ACOPSO pada *dataset* p_hat1500-3 dan DSJC1000_5.

IACOPSO membutuhkan jumlah iterasi yang relatif lebih banyak untuk mendapatkan maksimum *clique* dibangingkan dengan algoritma IACO dan ACOPSO Namun, masih lebih baik jika dibandingkan dengan ACO. Tabel lengkap dapat dilihat pada Lampiran 3. Walaupun IACOPSO lebih lambat penemuan *clique* dalam hal waktu, masih terbayarkan dengan hasil maksimum *clique* yang lebih baik dan standar deviasi yang relatif lebih rendah dibandingkan dengan ACO, ACOPSO dan IACO.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil pengujian yang telah dilakukan dapat ditarik kesimpulan sebagai berikut:

1. IACOPSO dapat menemukan maksimum *clique* yang lebih tinggi dibandingkan dengan algoritma lain dengan rata-rata hasil 40,42 untuk IACOPSO, 40,33 untuk ACOPSO, dan 40,06 untuk IACO. Sedangkan dari segi waktu, IACO lebih unggul dengan rata-rata waktu 25,557 detik untuk IACO, 25,797 detik untuk ACOPSO, dan 25,842 detik untuk IACOPSO.
2. IACOPSO dapat menemukan maksimum *clique* yang lebih konsisten dengan standar deviasi yang lebih rendah dibandingkan dengan algoritma ACO, IACO, dan ACOPSO. Akan tetapi, waktu yang dibutuhkan oleh IACOPSO untuk menemukan maksimum *clique* tidak konsisten jika dibandingkan dengan algoritma IACO dan ACOPSO. Hal ini ditunjukkan dengan standar deviasi waktu yang tinggi.
3. IACOPSO membutuhkan jumlah iterasi yang relatif lebih banyak namun hasil yang ditemukan masih lebih baik dibandingkan dengan algoritma ACO, IACO, dan ACOPSO.

5.2 Saran

Saran yang dapat diberikan untuk pengembangan selanjutnya adalah

1. Untuk mendapatkan hasil yang lebih baik, algoritma IACOPSO dapat ditambahkan dengan *local search* atau *pheromone* dapat ditempatkan di simpul untuk menurunkan waktu yang dibutuhkan dalam mencari maksimum *clique*.
2. Menggunakan metode lain untuk membantu IACOPSO agar tidak terjebak pada *local optimum* seperti metode yang dilakukan oleh Zhang pada pemilihan simpul kandidat. Zhang membuat tabel yang akan mencatat kemunculan setiap

simpul pada *clique*. Simpul yang terpilih adalah simpul dengan kemunculan terendah (Zhang et al., 2015).

DAFTAR PUSTAKA

Baglioni, M., Geraci, F., Pellegrini, M. & Lastres, E., 2014. Fast Exact and Approximate Computation. In Can, F., Özyer, T. & Polat, F. *State of the Art Applications of Social Network Analysis*. Springer International Publishing Switzerland. p.54.

Batagelj, V. & Mrvar, A., 2006. *Pajek Dataset*. [Online] Available at: http://vlado.fmf.uni-lj.si/pub/networks/data/ [Accessed 18 December 2015].

Battiti, R. & Protasi, M., 1995. Reactive Local Search for the Maximum Clique Problem. *International Computer Science Institute*.

Bomze, M.I., Budinich, M., Pardalos, M.P. & Pellio, M., 1999. The Maximum Clique Problem. In D.-Z. Du & M.P. Pardalos, eds. *Handbook of Combinatorial Optimization*. Volume A ed. Kluwer Academic Publisher. pp.41 - 47.

Brownlee, J., 2011. *Clever Algorithms: Nature-Inspired Programming Recipes*. 1st ed. Lulu.

Christakis, N.A. & Fowler, J.H., 2009. *Connected: The Amazing Power of Social Network and how They Shape Our Lives*. Hachette Book Group.

Deneubourg, J.L., Aron, S., Goss, S. & Pasteels, J.M., 1989. The Self-Organizing Exploratory Pattern of the Argentine Ant. *Journal of lnsect Behavior*, III.

Dorigo, M. & Stutzle, T., 2004. *Ant Colony Optimization*. The MIT Press.

Easley, D. & Kleinberg, J., 2010. Network, Crowds, and Markets: Reasoning about a Highly Connected World.

Kennedy, J. & Eberhart, R., 1995. Particle Swarm Optimization. *Institute of Electrical and Electronics Engineers*.

Kleinberg, J. & Tardos, E., 2006. *Algorithm Design*. Pearson Addison Wesley.

Krebs, E.V., 2002. Mapping Networks of Terrorist Cell. *INSNA*.

Leskovec, J., 2009. *Stanford Large Network Dataset Collection*. [Online] Available at: http://snap.stanford.edu/data/ [Accessed 18 December 2015].

Li, j. et al., 2007. A Multi-Stage Negotiation Mechanism with Service Oriented Implementation. In B. Akhgar, ed. *Proceedings of the 15th International Workshops on Conceptual Structures*. Springer. p.137.

Luger, G.F., 2005. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. 5th ed. Pearson Education Limited.

Mascia, F., 1993. *dimacs benchmark set*. [Online] Available at: http://iridia.ulb.ac.be/~fmascia/maximum_clique/DIMACS-benchmark [Accessed 18 December 2015].

Newman, M., 2013. *Network Data*. [Online] Available at: http://www-personal.umich.edu/~mejn/netdata/ [Accessed 18 December 2015].

Pattilo, J., Youssef, N. & Butenko, S., 2012. Clique Relaxation Models in Social Network Analysis. In M.P. Pardalos & M.T. Thai, eds. *Handbook of Optimization in Complex Networks : Communication and Social Networks*. 58th ed. Springer Science+ Business Media, LLC. pp.143 - 144.

Pouri, M.S., Rezvanian, A. & Meybodi, M.R., 2012. Finding a Maximum Clique Using Ant Colony Optimization and Particle Swarm Optimization in Social Media. In *ASONAM '12 Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*., 2012. ACM Digital Library.

Roberts, E. & Hennesy, J., 2003. *DNA Computing*. [Online] Available at: http://cs.stanford.edu/people/eroberts/courses/soco/projects/2003-04/dna-computing/clique.htm [Accessed 22 Desember 2015].

Russel, S. & Norvig, P., 1995. *Artificial Intelligence a Modern Approach*. 2nd ed. Pearson Education.

Solnon, C. & Fenet, S., 2006. A study of ACO capabilities for solving the maximum. *Springer Science+Business Media*.

Taani, A.-T.A. & Nemrawi, M.K., 2012. Solving the Maximum Clique Problem Using Intelligent Water Drops Algorithm. In *The International Conference on Computing, Networking and Digital Technologies (ICCNDT2012)*., 2012.

Wasserman, S. & Faust, K., 1994. *Social Network Analysis : Methods and Applications*. Cambridge University Press.

Wiil, U.K., Gniadek, J. & Memon, N., 2010. Measuring Link Importance in Terrorist Networks. *IEEE Computer Society*.

Wilson, R.J., 1996. *Introduction to Graph Theory*. 4th ed. Addison Wesley Longman Limited.

Xu, S., Ma, J. & Lei, J., 2007. An Improved Ant Colony Optimization for the Maximum Clique Problem. In *Third International Conference on Natural Computation*., 2007. IEEE.

Yan, F. et al., 2011. A Clique-Superstition Model for Social Network. *Science China Press*.

Yang, S.S., 2014. *Nature Inspired Optimization Algorithms*. 1st ed. Elsevier.

Zhang, S., Dong, Y., Yin, J. & Guo, J., 2015. Improved Ant Colony Algorithm for Finding the Maximum Clique in Social Network. In *IEEE 2nd International Conference on Cyber Security and Cloud Computing*., 2015. Institute of Electrical and Electronics Engineers.

-

Lampiran 1 *Listing Program*

Main.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Gabung_tugas_TA
{
    public partial class Main :
Form
    {
        public Main()
        {
            InitializeComponent();

        }
        private void
button1_Click(object sender,
EventArgs e)
        {
            Form1 f1 = new
Form1();
            f1.ShowDialog();
        }

        private void
button2_Click(object sender,
EventArgs e)
        {
            Form2 f2 = new
Form2();
            f2.ShowDialog();
        }
    }
}
```

Form 1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Diagnostics;

namespace Gabung_tugas_TA
{
    public partial class Form1 :
Form
    {
```

```
        enum algomode
        {
            ACO,
            IACO,
            ACOPSO,
            IACOPSO,
        };
        public static int
pengujian;
        public static List<string>
filename = new List<string>();
        public static List<string>
filepath = new List<string>();
        public Stopwatch watch =
new Stopwatch();
        public static int iseed =
3;
        public static int
bestcliquesize = 0;
        public static int alpha;
        public static double beta
= 0.05;
        public static List<int>
bestcliquecycle = new List<int>();
        public static List<int>
bestclique = new List<int>();
        public static List<int>
currentclique = new List<int>();
        public static int
currentcliquesize;
        public static int
bestcliquecyclesize = 0;
        public static int best =
0;
        public static double rho;
        public static double rhot;
        public static int
maxcycles;
        public static int nbants;
        public static double
taumin = 0.01;
        public static double
taumax = 6;
        public static int mode =
4;
        public bool check = false;
        public int countopen = 0;
        public Random rnd = new
Random();
        public static double T =
1.0;
        public static double
constpositive = 0.0002;

        public Form1()
        {
            InitializeComponent();
        }
        private void
Form1_Load(object sender,
EventArgs e)
        {
            maxcycles =
int.Parse(iterasinum.Text);
            nbants =
int.Parse(semutnum.Text);
            alpha =
int.Parse(alphanum.Text);
            rho =
double.Parse(rhonum.Text);
            rhot = rho;
            pengujian =
int.Parse(pengujiannum.Text);
        }

        public void runants(int
nbcycles, int algo, double qty)
        {
            for (nbcycles = 0;
((nbcycles < maxcycles) &&
(bestcliquesize != best));
nbcycles++)
            {
bestcliquecyclesize = 0;
                for (int i = 0;
((i < nbants) && (bestcliquesize
!= best)); i++)
                {
                    if (algo == 1
|| algo == 2 || algo == 3)
                    {
currentcliquesize =
Ants.walkE(alpha,
rand.getnextrand(graph.nbvertices)
, algo, 1);
                    }
                    else
                    {
currentcliquesize =
Ants.walkE(alpha,
rand.getnextrand(graph.nbvertices)
, algo, 1);
                    }
                    if
(currentcliquesize >
bestcliquecyclesize)
                    {
bestcliquecycle.Clear();
                        for (int j
= 0; j < currentcliquesize; j++)
                        {
bestcliquecycle.Add(currentclique[
j]);
                        }
```

```
bestcliquecyclesize =
currentcliquesize;
                    }
                }
                if
(bestcliquecyclesize >
bestcliquesize)
                {
                    bestcliquesize
= bestcliquecyclesize;
bestclique.Clear();
                    for (int j =
0; j < bestcliquesize; j++)
bestclique.Add(bestcliquecycle[j])
;
                    if
(bestcliquesize >= best)
                    {
bestcliquesize = best;
                    }
                }
                graph.evaporate(1
- rho, taumin);
                if (nbcycles <=
100)
                {
                    alpha = 1;
                }
                else if (nbcycles
> 100 && nbcycles <= 400)
                {
                    alpha = 2;
                }
                else if (nbcycles
> 400 && nbcycles <= 800)
                {
                    alpha = 3;
                }
                else
                {
                    alpha = 4;
                }

                if (rho <= 1)
                {
                    rho = (1 -
constpositive) * rho;
                }
                else if (rho <= 0)
                {
                    rho = 0.001;
                }

                if (algo == 0 ||
algo == 1)
                {
                    qty = 1.0 /
(double)(1 + bestcliquesize -
bestcliquecyclesize);
                }
graph.reinforcement(bestcliquecycl
esize, taumax, qty, algo, 1);
            }
        }

        public void runpengujian()
        {
            DateTime dt =
DateTime.Now;
            currentcliquesize = 0;
            double qty = 0;
            int nbcycles = 0;
            int progress = 1;
            foreach (int item in
checkedListBox1.CheckedIndices)
            {
                DataGridView
dataGridView1 = new
DataGridView();
                TabPage tabpage =
new TabPage();
                tabpage.Text =
Path.GetFileNameWithoutExtension(f
ilename[item]);
graph.succ.Clear();
graph.readfiles(filepath[item]);
                rand.seed(iseed);
graph.creategraph();
                if (graph.cek ==
false)
                {
MessageBox.Show("Kesalahan dalam
pembacaan data. mohon cek data
apakah sesuai dengan format
DIMACS? atau simpul dimulai dari
1? jika tidak mohon untuk
mengganti formatnya" +
filename[item]);
                    return;
                }
graph.initpheromone(taumax,1);
                best =
graph.nbvertices;
                for (int algo = 0;
algo < mode; algo++)
                {
                    algomode value
= new algomode();
                    if (algo == 0)
                        value =
algomode.ACO;
                    else if (algo
== 1)
```

```
                    value =
algomode.IACO;
                else if (algo
== 2)
                    value =
algomode.ACOPSO;
                else if (algo
== 3)
                    value =
algomode.IACOPSO;
                for (int k =
0; k < pengujian; k++)
                {
backgroundWorker1.ReportProgress(p
rogress);
progress++;
bestcliquesize = 0;
                    T = 1.0;
graph.deltatau = 0;
                    graph.v =
0;
                    qty = 0;
                    rho =
rhot;
bestclique.Clear();
bestcliquecycle.Clear();
Ants.p.Clear();
currentclique.Clear();
watch.Reset();
GC.Collect();
watch.Start();
runants(nbcycles, algo, qty);
watch.Stop();
                    if
(bestcliquesize <= 2)
                    {
bestcliquesize = 0;
bestclique.Clear();
                    }
dataGridView1.Dock =
DockStyle.Fill;
dataGridView1.ColumnCount = 6;
dataGridView1.Columns[0].Name =
"Nama Dataset";
dataGridView1.Columns[1].Name =
"Simpul";
dataGridView1.Columns[1].Width =
70;
dataGridView1.Columns[2].Name =
"Sisi";
dataGridView1.Columns[2].Width =
70;
dataGridView1.Columns[3].Name =
"Algoritma";
dataGridView1.Columns[4].Name =
"Best";
dataGridView1.Columns[5].Name =
"Time";
dataGridView1.Rows.Add(Path.GetFil
eNameWithoutExtension(filename[ite
m]), graph.nbvertices,
graph.nbedge, value,
Convert.ToString(bestcliquesize),
watch.Elapsed);
dataGridView1.ReadOnly = true;
dataGridView1.AllowUserToAddRows =
false;
dataGridView1.Refresh();
                    using
(FileStream fs = new
FileStream("Hasil_Pengujian.txt",
FileMode.Append,
FileAccess.Write))
                    using
(StreamWriter w = new
StreamWriter(fs))
                    {
w.WriteLine(dt.ToShortDateString()
+ ";" +
Path.GetFileNameWithoutExtension(f
ilename[item]) + ";" +
graph.nbvertices + ";" +
graph.nbedge + ";" + alpha + ";" +
rho + ";" + value + ";" +
Convert.ToString(bestcliquesize) +
";" + watch.Elapsed);
                    }
graph.refreshtaue();
GC.Collect();
                }
                GC.Collect();
            }
Array.Clear(graph.matrix, 0,
graph.nbvertices);
Array.Clear(graph.tauE, 0,
graph.nbvertices);
Array.Clear(graph.tauEinitial, 0,
graph.nbvertices);
tabpage.Controls.Add(dataGridView1
);
Addtabpage(tabpage);
            using (FileStream
fs = new
FileStream("Hasil_Pengujian.txt",
FileMode.Append,
FileAccess.Write))
            using
(StreamWriter w = new
StreamWriter(fs))
            {
                w.WriteLine();
            }
        }
```

```
        }

        private void
open_Click(object sender,
EventArgs e)
        {
            OpenFileDialog
openfile = new OpenFileDialog();
            openfile.Filter =
"Text Files (.txt)|*.txt|DIMACS
Benchmark (.clq)|*.clq|All
Files(*.*)|*.*";
            openfile.FilterIndex =
1;
            openfile.Multiselect =
true;
            if
(openfile.ShowDialog() ==
DialogResult.OK)
            {
                foreach (string
var in openfile.SafeFileNames)
                {
checkedListBox1.Items.Add(var,
CheckState.Checked);
filename.Add(var);
                }
                foreach (string
var in openfile.FileNames)
                {
filepath.Add(var);
                }
                Mulai.Enabled =
true;
                Clear.Enabled =
true;
                Mulai.BackColor =
Color.LimeGreen;
                Clear.Enabled =
true;
                check = true;
            }

        }
        private void
Editpara_Click(object sender,
EventArgs e)
        {
            Editpara.Enabled =
false;
            Mulai.Enabled = false;
            Mulai.BackColor =
SystemColors.Control;
            okpara.Enabled = true;
            pengujiannum.Enabled =
true;
            iterasinum.Enabled =
true;
            semutnum.Enabled =
true;
            alphanum.Enabled =
true;
            rhonum.Enabled = true;
        }
        private void
okpara_Click(object sender,
EventArgs e)
        {
            iterasinum.Enabled =
false;
            semutnum.Enabled =
false;
            alphanum.Enabled =
false;
            rhonum.Enabled =
false;
            pengujiannum.Enabled =
false;
            try
            {
                if
(int.Parse(iterasinum.Text) < 1 ||
int.Parse(semutnum.Text) < 1 ||
int.Parse(pengujiannum.Text) < 1)
                {
MessageBox.Show("Jumlah iterasi
atau jumlah semut atau jumlah
pengujian minimal 1 ");
                }
                else if
(double.Parse(alphanum.Text) < 0
|| double.Parse(rhonum.Text) < 0)
                {
MessageBox.Show("Jumlah iterasi
atau jumlah semut atau jumlah
pengujian minimal 0 ");
                }
                else
                {
                    maxcycles =
int.Parse(iterasinum.Text);
                    nbants =
int.Parse(semutnum.Text);
                    alpha =
int.Parse(alphanum.Text);
                    rho =
double.Parse(rhonum.Text);
                    rhot = rho;
                    pengujian =
int.Parse(pengujiannum.Text);
Editpara.Enabled = true;
                }
            }
```

```
            catch (Exception)
            {
MessageBox.Show("Parameter yang
dimasukan harus berupa angka atau
ada parameter yang tidak sesuai
(jumlah iterasi, jumlah semut, dan
jumlah pengujian tidak boleh angka
KOMA!");
                Editpara.Enabled =
true;
            }
            if (check == true)
            {
                Mulai.Enabled =
true;
                Mulai.BackColor =
Color.LimeGreen;
            }
            okpara.Enabled =
false;
        }
        private void
Mulai_Click(object sender,
EventArgs e)
        {
            countopen =
checkedListBox1.CheckedIndices.Cou
nt;
            progressBar1.Maximum =
(countopen* 4* pengujian);
            Mulai.Enabled = false;
            Clear.Enabled = false;
            Mulai.BackColor =
SystemColors.Control;
            open.Enabled = false;
checkedListBox1.Enabled = false;
            Clear.Enabled = false;
            Editpara.Enabled =
false;
tabControl1.TabPages.Clear();
            List<string> rows =
new List<string>();
            bestclique.Clear();
bestcliquecycle.Clear();
            graph.succ.Clear();
            Ants.p.Clear();
            currentclique.Clear();
backgroundWorker1.RunWorkerAsync()
;
        }
        private void
Clear_Click(object sender,
EventArgs e)
        {
checkedListBox1.Items.Clear();
            filename.Clear();
            filepath.Clear();
            Mulai.Enabled = false;
            Mulai.BackColor =
SystemColors.Control;
            Clear.Enabled = false;
        }
        private void
backgroundWorker1_DoWork(object
sender, DoWorkEventArgs e)
        {
            runpengujian();
        }
        private void
backgroundWorker1_ProgressChanged(
object sender,
ProgressChangedEventArgs e)
        {
            progressBar1.Value =
e.ProgressPercentage;
        }
        private void
backgroundWorker1_RunWorkerComplet
ed(object sender,
RunWorkerCompletedEventArgs e)
        {
MessageBox.Show("Proses Selesai");
            Editpara.Enabled =
true;
            open.Enabled = true;
checkedListBox1.Enabled = true;
            Mulai.Enabled = true;
            Mulai.BackColor =
Color.LimeGreen;
            Clear.Enabled = true;
        }
        public void
Addtabpage(TabPage tab)
        {
            if
(tabControl1.InvokeRequired)
            {
tabControl1.Invoke(new
Action<TabPage>(Addtabpage), tab);
                return;
            }
tabControl1.Controls.Add(tab);
        }
    }
}
```

Form 2.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Windows.Forms;
using System.Diagnostics;

namespace Gabung_tugas_TA
{
    public partial class Form2 :
Form
    {
        enum algomode
        {
            ACO,
            IACO,
            ACOPSO,
            IACOPSO,
        };
        public Stopwatch watch =
new Stopwatch();
        public static int iseed =
3;
        public static int
bestcliquesize = 0;
        public static Random rnd =
new Random();
        public static int alpha;
        public static double beta
= 0.05;
        public static List<int>
bestcliquecycle = new List<int>();
        public static List<int>
bestclique = new List<int>();
        public static List<int>
currentclique = new List<int>();
//{ get; set; }
        public List<List<object>>
Matriks;
        public static int best =
0;
        public static double rho;
        public static int
maxcycles;
        public static int nbants;
        public static double
taumin = 0.01;
        public static double
taumax = 6;
        public int selectedindex =
-1;
        public static int table =
0;
        public static int count =
0;
        public static string
algoritma;
        public static int
editcount = 0;
        public static double T =
1.0;
        public static double
constpositive = 0.0002;
        public Form2()
        {
            InitializeComponent();
        }
        private void
Form1_Load(object sender,
EventArgs e)
        {
comboBox1.Items.Add("Ant Colony
Optimization");
comboBox1.Items.Add("Improved Ant
Colony Optimization");
comboBox1.Items.Add("Ant Colony
Optimization + Particle Swarm
Optimization");
comboBox1.Items.Add("Improved Ant
Colony Optimization + Particle
Swarm Optimization");
            maxcycles =
int.Parse(iterasinum.Text);
            nbants =
int.Parse(semutnum.Text);
            alpha =
int.Parse(alphanum.Text);
            rho =
double.Parse(rhonum.Text);
dataGridView1.ColumnCount = 5;
dataGridView1.Columns[0].Name =
"Simpul";
dataGridView1.Columns[1].Name =
"Sisi";
dataGridView1.Columns[2].Name =
"Algoritma";
dataGridView1.Columns[3].Name =
"Best";
dataGridView1.Columns[4].Name =
"Time";
        }
```

```
        private void
Editpara_Click(object sender,
EventArgs e)
        {
            Editpara.Enabled =
false;
            Mulai.Enabled = false;
            Mulai.BackColor =
Color.Empty;
            okpara.Enabled = true;
            iterasinum.Enabled =
true;
            semutnum.Enabled =
true;
            alphanum.Enabled =
true;
            rhonum.Enabled = true;
        }
        private void
okpara_Click(object sender,
EventArgs e)
        {
            if(editcount==1)
            {
                Mulai.Enabled =
true;
            }
            Editpara.Enabled =
true;
            okpara.Enabled =
false;
            iterasinum.Enabled =
false;
            semutnum.Enabled =
false;
            alphanum.Enabled =
false;
            rhonum.Enabled =
false;
        }

        private void
Mulai_Click(object sender,
EventArgs e)
        {
            double qty;
            int i, j;
            int nbcycles;
            int nbiter = 0;
            int currentcliquesize;
            int
bestcliquecyclesize = 0;
            List<string> rows =
new List<string>();
dataGridView1.Rows.Clear();
dataGridView1.Refresh();
            bestclique.Clear();
bestcliquecycle.Clear();
            graph.succ.Clear();
            Ants.p.Clear();
            currentclique.Clear();
            bestcliquesize = 0;
            bestclique.Clear();
bestcliquecycle.Clear();
            graph.succ.Clear();
            Ants.p.Clear();
            currentclique.Clear();
            bestcliquesize = 0;
            rand.seed(iseed);
graph.creategraphtable(table);
graph.initpheromone(taumax,2);
            best =
graph.nbvertices;
            algomode value = new
algomode();
            if (selectedindex ==
0)
            {
                value =
algomode.ACO;
                algoritma = "Ant
Colony Optimization";
            }
            else if (selectedindex
== 1)
            {
                value =
algomode.IACO;
                algoritma =
"Improved Ant Colony
Optimization";
            }
            else if (selectedindex
== 2)
            {
                value =
algomode.ACOPSO;
                algoritma = "Ant
Colony Optimization + Particle
Swarm Optimization";
            }
            else if (selectedindex
== 3)
            {
                value =
algomode.IACOPSO;
                algoritma =
"Improved Ant Colony Optimization
+ Particle Swarm Optimization";
            }
            watch.Reset();
            watch.Start();
            for (nbcycles = 0;
((nbcycles < maxcycles) &&
```

```
(bestcliquesize < best));
nbcycles++)
            {
bestcliquecyclesize = 0;
                for (i = 0; ((i <
nbants) && (bestcliquesize <
best)); i++)
                {
currentcliquesize =
Ants.walkE(alpha,
rand.getnextrand(graph.nbvertices)
, selectedindex,2);
                    nbiter =
currentcliquesize;
                    if
(currentcliquesize >
bestcliquecyclesize)
                    {
bestcliquecycle.Clear();
                        for (j =
0; j < currentcliquesize; j++)
                        {
bestcliquecycle.Add(currentclique[
j]);
                        }
bestcliquecyclesize =
currentcliquesize;
                    }
                }
                if
(bestcliquecyclesize >
bestcliquesize)
                {
                    bestcliquesize
= bestcliquecyclesize;
bestclique.Clear();
                    for (j = 0; j
< bestcliquesize; j++)
bestclique.Add(bestcliquecycle[j])
;
                    if
(bestcliquesize >= best)
                    {
bestcliquesize = best;
                    }
                }
                graph.evaporate(1
- rho, taumin);
                if (nbcycles <=
100)
                {
                    alpha = 1;
                }
                else if (nbcycles
> 100 && nbcycles <= 400)
                {
                    alpha = 2;
                }
                else if (nbcycles
> 400 && nbcycles <= 800)
                {
                    alpha = 3;
                }
                else
                {
                    alpha = 4;
                }

                if (rho <= 1)
                {
                    rho = (1 -
constpositive) * rho;
                }
                else if (rho <= 0)
                {
                    rho = 0.001;
                }
                qty = 1.0 /
(double)(bestcliquesize + 1 -
bestcliquecyclesize);
graph.reinforcement(bestcliquecycl
esize, taumax, qty,
selectedindex,2);
            }
            watch.Stop();
            if (bestcliquesize <=
2)
            {
                bestcliquesize =
0;
bestclique.Clear();
            }
dataGridView1.Rows.Add(graph.nbver
tices, graph.nbedgep, value,
Convert.ToString(bestcliquesize),
watch.Elapsed);
dataGridView1.Refresh();
Array.Clear(graph.tauE, 0,
graph.nbvertices);
            Ulang.Enabled = true;
            T_gmbar.Enabled =
true;
            Editpara.Enabled =
false;
            Mulai.Enabled = false;
        }
        private void
comboBox1_SelectedIndexChanged(obj
ect sender, EventArgs e)
        {
            Pilih.Enabled = true;
        }
```

```
        private void
Pilih_Click(object sender,
EventArgs e)
        {
            selectedindex =
comboBox1.SelectedIndex;
            J_node.Enabled = true;
            b_node.Enabled = true;
            Pilih.Enabled = false;
        }

        private void
button1_Click(object sender,
EventArgs e)
        {
            comboBox1.Enabled =
false;
            b_node.Enabled =
false;
            J_node.Enabled =
false;
            S_button.Enabled =
true;
            B_button.Enabled =
true;
Table_grid.Rows.Clear();
Table_grid.Columns.Clear();
            DataGridViewRow rows =
new DataGridViewRow();
            DataGridViewColumn
columns = new
DataGridViewColumn();
            table =
Convert.ToInt32(J_node.Text);
            Table_grid.ColumnCount
= table;
            Table_grid.RowCount =
table;
Table_grid.RowHeadersWidthSizeMode
=
DataGridViewRowHeadersWidthSizeMod
e.DisableResizing;
Table_grid.ColumnHeadersHeightSize
Mode =
DataGridViewColumnHeadersHeightSiz
eMode.DisableResizing;
            for (int i = 0; i <
table; i++)
            {
                rows =
Table_grid.Rows[i];
                rows.Height = 25;
                columns =
Table_grid.Columns[i];
                columns.Width =
25;
                for (int j = 0; j
< table; j++)
                {
Table_grid.Columns[j].HeaderText =
(j).ToString();
Table_grid.Rows[i].HeaderCell.Valu
e = (i).ToString();
Table_grid.Rows[i].Cells[j].ReadOn
ly = true;
Table_grid.Rows[i].Cells[j].Value
= 0;
                    if (i == j)
                    {
Table_grid.Rows[i].Cells[j].Style.
BackColor = Color.Red;
                    }
                }
            }
        }

        private void
Table_grid_CellClick(object
sender, DataGridViewCellEventArgs
e)
        {
            try {
                int value =
int.Parse(Table_grid.Rows[e.RowInd
ex].Cells[e.ColumnIndex].Value.ToS
tring());
                if (e.RowIndex !=
e.ColumnIndex)
                {
                    if (value ==
0)
                    {
Table_grid.Rows[e.RowIndex].Cells[
e.ColumnIndex].Value = 1;
Table_grid.Rows[e.ColumnIndex].Cel
ls[e.RowIndex].Value = 1;
                    }
                    else
                    {
Table_grid.Rows[e.RowIndex].Cells[
e.ColumnIndex].Value = 0;
Table_grid.Rows[e.ColumnIndex].Cel
ls[e.RowIndex].Value = 0;
                    }
                }
            }
            catch
            {
MessageBox.Show("Jangan Klik
Header Table");
            }
        }
```

```
        private void
S_button_Click(object sender,
EventArgs e)
        {
            Matriks = new
List<List<object>>();
            for (int k = 0; k <
table; k++)
            {
                List<object>
submatrix = new List<object>();
                for (int l = 0; l
< table; l++)
                {
submatrix.Add(Table_grid.Rows[k].C
ells[l].Value);
                }
graph.matrixp.Add(submatrix);
Gambar.matp.Add(submatrix);
Matriks.Add(submatrix);
            }
            S_button.Enabled =
false;
            B_button.Enabled =
false;
            Mulai.Enabled = true;
            Table_grid.Enabled =
false;
            editcount = 1;
        }

        private void
B_button_Click(object sender,
EventArgs e)
        {
            comboBox1.Enabled =
true;
            B_button.Enabled =
false;
            S_button.Enabled =
false;
            selectedindex = -1;
Table_grid.Rows.Clear();
Table_grid.Columns.Clear();
        }

        private void
T_gmbar_Click(object sender,
EventArgs e)
        {
            Gambar gmbr = new
Gambar();
            gmbr.Show();
        }
```

```
        private void
Ulang_Click(object sender,
EventArgs e)
        {
            comboBox1.Enabled =
true;
            Table_grid.Enabled =
true;
            Ulang.Enabled = false;
            Editpara.Enabled =
true;
            T_gmbar.Enabled =
false;
            editcount = 0;
Table_grid.Rows.Clear();
Table_grid.Columns.Clear();
            Table_grid.Refresh();
dataGridView1.Rows.Clear();
dataGridView1.Refresh();
            graph.matrixp.Clear();
            Gambar.matp.Clear();
            table = 0;
bestcliquecycle.Clear();
            bestcliquesize = 0;
Form2.currentclique.Clear();
            count = 0;
            Matriks.Clear();
        }

    }
}
```

Ants.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Gabung_tugas_TA
{
    class Ants
    {
        public static List<double>
p = new List<double>();
        public static Random rnd =
new Random();
        public static double
mypow(double x,int y)
        {
            double P;
            if (y == 0)
                return 1;
            else if (y == 1)
                return x;
            else if (y == 2)
                return x * x;
            else
            {
                if (y % 2 == 0)
                {
                    P = mypow(x, y
/ 2);
                    return P * P;
                }
                else
                {
                    P = mypow(x,
(y - 1) / 2);
                    return P * P *
x;
                }
            }
        }

        public static int
choose(int nbcand, int algo,int
cek)
        {
            double f =
rnd.NextDouble();
            int left = 0;
            int right = nbcand -
1;
            int k = 0;
            double total =
p[nbcand - 1];
            while (left < right)
            {
                k = (left + right
+ 1) / 2;
                if (algo == 0 ||
algo == 2)
                {
                    if (f < p[k -
1] / total)
                        right = k
- 1;
                    else if (f >
p[k] / total)
                        left = k +
1;
                    else
                        return k;
                }
                else if (algo == 1
|| algo == 3)
                {
                    if (cek == 1)
                    {
                        if (f <
(p[k - 1] / total) + (Form1.T *
(graph.degree[k - 1] /
graph.nbedge)))
                            right
= k - 1;
                        else if (f
> (p[k] / total) + (Form1.T *
(graph.degree[k] / graph.nbedge)))
                            left =
k + 1;
                        else
                            return
k;
                        Form1.T =
Form1.T * (1 - Form1.beta);
                    }
                    else
                    {
                        if (f <
(p[k - 1] / total) + (Form2.T *
(graph.degree[k - 1] /
graph.nbedgep)))
                            right
= k - 1;
                        else if (f
> (p[k] / total) + (Form2.T *
(graph.degree[k] /
graph.nbedgep)))
                            left =
k + 1;
```

```
                        else
                            return
k;
                        Form2.T =
Form2.T * (1 - Form2.beta);
                    }
                }
            }
            if (left >= 0 && left
< nbcand)
                return left;
            else
                return k;
        }
        public static int
walkE(int alpha, int firstvertex,
int algo, int cek)
        {
            int i, v;
            List<int> candidates =
new List<int>();
            if (cek == 1)
            {
Form1.currentclique = new
List<int>();
            }
            else
            {
Form2.currentclique = new
List<int>();
            }
            int nbcandidates;
            double total;
            double[] taoclique =
new double[graph.nbvertices];
            int cliquesize = 1;
            if (cek == 1)
            {
Form1.currentclique.Add(firstverte
x);
            }
            else
            {
Form2.currentclique.Add(firstverte
x);
            }
            nbcandidates =
graph.degree[firstvertex];
            p.Clear();
            total = 0;
            for (i = 0; i <
nbcandidates; i++)
            {
candidates.Add(graph.succ[firstver
tex][i]);
taoclique[candidates[i]] =
graph.tauE[firstvertex,
candidates[i]];
p.Add(mypow(taoclique[candidates[i
]], alpha)+total);
                total = p[i];
            }
            while (nbcandidates >
0)
            {
                i =
choose(nbcandidates, algo, cek);
                v = candidates[i];
                if (cek == 1)
                {
Form1.currentclique.Add(v);
                }
                else
                {
Form2.currentclique.Add(v);
                }
                cliquesize++;
candidates.Remove(candidates[i]);
                nbcandidates--;
                p.Clear();
                i = 0;
                total = 0;
                while (i <
nbcandidates)
                {
                    if
(graph.tauE[v, candidates[i]] > 0)
                    {
taoclique[candidates[i]] =
graph.tauE[firstvertex,
candidates[i]];
p.Add(mypow(taoclique[candidates[i
]], alpha) + total);
                        total =
p[i];
                        i++;
                    }
                    else
                    {
nbcandidates--;
candidates.Remove(candidates[i]);
                    }
                }
            }
            Array.Clear(taoclique,
0, graph.nbvertices);
            return cliquesize;
        }
    }
}
```

Graph.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Linq;
using System.Text;
using System.IO;

namespace Gabung_tugas_TA
{
    class graph
    {
        public static int
nbvertices;
        public static int nbedge;
        public static int nbedgep;
        public static double[,]
tauE;
        public static double[,]
tauEinitial;
        public static int[]
degree;
        public static
List<List<int>> succ = new
List<List<int>>();
        public static int[,]
matrix;
        public static
List<List<object>> matrixp = new
List<List<object>>();
        public static bool cek =
true;
        public static int cekgraph
= 0;
        public static int cekE =
0;
        public static double v =
0;
        public static double
deltatau = 0;
        public static double c1 =
0.3, c2 = 1-c1, c3 = 0.3;
        public static Random rnd =
new Random();

        public static void
readfiles(string graphfile)
        {
            cekE = 0;
            cekgraph = 0;
            FileStream ifs = new
FileStream(graphfile,
FileMode.Open);
            StreamReader sr = new
StreamReader(ifs);
            string line = "";
            string[] tokens =
null;
            line = sr.ReadLine();
            line = line.Trim();
            try
            {
                while (line !=
null && line.StartsWith("p") ==
false)
                {
                    line =
sr.ReadLine();
                    line =
line.Trim();
                }
                tokens =
line.Split(' ');
                nbvertices =
int.Parse(tokens[2]);
                nbedge =
int.Parse(tokens[3]);
                matrix = new
int[nbvertices, nbvertices];
                while ((line =
sr.ReadLine()) != null)
                {
                    line =
line.Trim(' ');
                    if
(line.StartsWith("e") == true)
                    {
                        tokens =
line.Split(' ');
                        int nodeA
= int.Parse(tokens[1]) - 1;
                        int nodeB
= int.Parse(tokens[2]) - 1;
matrix[nodeA, nodeB] = 1;
matrix[nodeB, nodeA] = 1;
                        cekE = 1;
                    }
                    else if
(line.StartsWith("c") ||
line.StartsWith("#") ||
line.StartsWith("*") == true)
                    {
                    }
                    else if
(line.StartsWith("0") == true &&
cekE != 1)
                    {
```

```
                        tokens =
line.Split(' ');
                        int nodeA
= int.Parse(tokens[0]);
                        int nodeB
= int.Parse(tokens[1]);
matrix[nodeA, nodeB] = 1;
matrix[nodeB, nodeA] = 1;
                        cekgraph =
1;
                    }
                    else if
(cekgraph == 1 && cekE != 1)
                    {
                        if
((Convert.ToInt32(line.StartsWith(
"1")) >= 49 ||
Convert.ToInt32(line.StartsWith("9
")) <= 57) == true)
                        {
                            tokens
= line.Split(' ');
                            int
nodeA = int.Parse(tokens[0]);
                            int
nodeB = int.Parse(tokens[1]);
matrix[nodeA, nodeB] = 1;
matrix[nodeB, nodeA] = 1;
                        }
                    }
                    else if
(cekgraph == 0 && cekE == 0)
                    {
                        if
((Convert.ToInt32(line.StartsWith(
"1")) >= 48 ||
Convert.ToInt32(line.StartsWith("9
")) <= 57) == true)
                        {
                            tokens
= line.Split(' ');
                            int
nodeA = int.Parse(tokens[0]) - 1;
                            int
nodeB = int.Parse(tokens[1]) - 1;
matrix[nodeA, nodeB] = 1;
matrix[nodeB, nodeA] = 1;
                        }
                    }

                }
            }
            catch (Exception)
            {
                cek = false;
            }
            sr.Close();
                ifs.Close();
        }

        public static void
refreshtaue()
        {
            for (int i = 0; i <
nbvertices; i++)
            {
                for (int j = 0; j
< nbvertices; j++)
                {
                    tauE[i, j] =
tauEinitial[i, j];
                    tauE[j, i] =
tauEinitial[j, i];
                }
            }
        }
        public static void
creategraph()
        {
            tauE = new
double[nbvertices, nbvertices];
            tauEinitial = new
double[nbvertices, nbvertices];
            degree = new
int[nbvertices];
            List<double> subtauE =
new List<double>();
            int nbsucc;
            for (int i = 0; i <
nbvertices; i++)
            {
                for (int j = 0; j
< nbvertices; j++)
                {
                    if
(matrix[i,j] == 1)
                    {
                        tauE[i, j]
= 1.0;
                        tauE[j, i]
= 1.0;
tauEinitial[i, j] = 1.0;
tauEinitial[j, i] = 1.0;
degree[i]++;
                    }
                }
            }
            for (int i = 0; i <
nbvertices; i++)
            {
                List<int> subsucc
= new List<int>();
                nbsucc = 0;
                int j = 0;
```

```
                for (j = 0; j <
nbvertices; j++)
                {
                    if (tauE[i, j]
> 0)
                    {
subsucc.Add(j);
                        nbsucc++;
                    }
                }
                succ.Add(subsucc);
            }
        }
        public static void
createGraphTable(int row)
        {
            nbvertices =
Form2.table;
            tauE = new
double[nbvertices, nbvertices];
            degree = new
int[nbvertices];
            List<double> subtauE =
new List<double>();
            int nbsucc;
            for (int i = 0; i <
nbvertices; i++)
            {
                for (int j = 0; j
< nbvertices; j++)
                {
                    if
(Convert.ToInt32(matrixp[i][j]) ==
1)
                    {
                        tauE[i, j]
= 1.0;
                        tauE[j, i]
= 1.0;
degree[i]++;
                        nbedgep++;
                    }
                }
            }
            nbedgep = nbedgep / 2;
            for (int i = 0; i <
nbvertices; i++)
            {
                List<int> subsucc
= new List<int>();
                nbsucc = 0;
                int j = 0;
                for (j = 0; j <
nbvertices; j++)
                {
                    if (tauE[i, j]
> 0)
                    {
subsucc.Add(j);
                        nbsucc++;
                    }
                }
                succ.Add(subsucc);
            }
        }
        public static void
initpheromone(double taumax, int
cek)
        {
            int i, j;
            for (i = 0; i <
nbvertices; i++)
            {
                for (j = 0; j <
degree[i]; j++)
                {
                    tauE[i,
succ[i][j]] = taumax;
                    if (cek == 1)
                    {
tauEinitial[i, succ[i][j]] =
taumax;
                    }
                }
            }
        }
        public static void
evaporate(double rho, double
taumin)
        {
            for (int i = 0; i <
nbvertices; i++)
            {
                for (int j = 0; j
< degree[i]; j++)
                {
                    tauE[i,
succ[i][j]] *= rho;
                    if (tauE[i,
succ[i][j]] < taumin)
                    {
                        tauE[i,
succ[i][j]] = taumin;
                    }
                }
            }
        }
        public static void
reinforcement(int cliquesize,
double taumax, double qty, int
algo, int cek)
        {
            for (int i = 0; i <
cliquesize; i++)
```

```
            {
                for (int j = i +
1; j < cliquesize; j++)
                {
                    if (algo == 0
|| algo == 1)
                    {
                        if (cek ==
1)
                        {
tauE[Form1.bestcliquecycle[i],
Form1.bestcliquecycle[j]] += qty;
                        }
                        else
                        {
tauE[Form2.bestcliquecycle[i],
Form2.bestcliquecycle[j]] += qty;
                        }
                    }
                    else if (algo
== 2 || algo == 3)
                    {
                        if (cek ==
1)
                        {
                            v = c1
* rnd.NextDouble() *
(tauE[Form1.bestcliquecycle[i],
Form1.bestcliquecycle[j]] -
deltatau) + c2 * rnd.NextDouble()
* (tauE[Form1.bestclique[i],
Form1.bestclique[j]] - deltatau) +
c3 * v;
deltatau += v;
tauE[Form1.bestcliquecycle[i],
Form1.bestcliquecycle[j]] +=
deltatau;
                        }
                        else
                        {
                            v = c1
* rnd.NextDouble() *
(tauE[Form2.bestcliquecycle[i],
Form2.bestcliquecycle[j]] -
deltatau) + c2 * rnd.NextDouble()
* (tauE[Form2.bestclique[i],
Form2.bestclique[j]] - deltatau) +
c3 * v;
deltatau += v;
tauE[Form2.bestcliquecycle[i],
Form2.bestcliquecycle[j]] +=
deltatau;
                        }
                    }
                    if (cek == 1)
                    {
                        if
(tauE[Form1.bestcliquecycle[i],
Form1.bestcliquecycle[j]] >
taumax)
tauE[Form1.bestcliquecycle[i],
Form1.bestcliquecycle[j]] =
taumax;
tauE[Form1.bestcliquecycle[j],
Form1.bestcliquecycle[i]] =
tauE[Form1.bestcliquecycle[i],
Form1.bestcliquecycle[j]];
                        }
                        else
                        {
                            if
(tauE[Form2.bestcliquecycle[i],
Form2.bestcliquecycle[j]] >
taumax)
tauE[Form2.bestcliquecycle[i],
Form2.bestcliquecycle[j]] =
taumax;
tauE[Form2.bestcliquecycle[j],
Form2.bestcliquecycle[i]] =
tauE[Form2.bestcliquecycle[i],
Form2.bestcliquecycle[j]];
                        }
                    }
                }
            }
        }
    }
}
```

Rand.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Gabung_tugas_TA
{
    class rand
    {
       public static int iseed;
        public static double
randfloat()
        {
            const int ia = 16807,
ic = 2147483647, iq = 127773, ir =
2836;
            int il, ih, it;
            double rc;
            ih = iseed / iq;
            il = iseed % iq;
            it = ia * il - ir *
ih;
            if (it > 0)
                iseed = it;
            else
                iseed = ic + it;
            rc = ic;
            return (iseed / rc);
        }
        public static int
getnextrand(int limit)
        {
return(int)(100000*randfloat())%li
mit;
        }
        public static void
seed(int seed)
        {
            iseed = seed;
        }
    }
}
```

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace Gabung_tugas_TA
{
    static class Program
    {
        [STAThread]
        static void Main()
        {
```

```
Application.EnableVisualStyles();
Application.SetCompatibleTextRende
ringDefault(false);
            Application.Run(new
Main());
        }
    }
}
```

Lampiran 2 Hasil Lengkap Pencarian *Maximum Clique* Dengan ACO, IACO, ACOPSO & IACOPSO

Hasil	Best Known	ACO					IACO					ACOPSO					IACOPSO				
		Best	*Min*	*Std dev*	*Avg best*	*Avgtime*	*Best*	*Min*	*Std dev*	*Avg best*	*Avgtime*	*Best*	*Min*	*Std dev*	*Avg best*	*Avgtime*	*Best*	*Min*	*Std dev*	*Avg best*	*Avgtime*
Co-authorship of scientist in network theory and experiment	20,00	20	20	0	20,00	00:00,618	20	20	0	20,00	00:00,598	20	20	0	20,00	00:00,609	20	20	0	20,00	00:00,621
Dolphin network	5,00	5	5	0	5,00	00:00,070	5	5	0	5,00	00:00,058	5	5	0	5,00	00:00,059	5	5	0	5,00	00:00,059
email interchange network univ of rovia i virgili tarragon	12,00	12	12	0	12,00	00:00,886	12	12	0	12,00	00:00,865	12	12	0	12,00	00:00,864	12	12	0	12,00	00:00,870
Erdos991	7,00	7	7	0	7,00	00:00,228	7	7	0	7,00	00:00,215	7	7	0	7,00	00:00,217	7	7	0	7,00	00:00,219
Facebook_combined	69,00	69	69	0	69,00	00:11,955	69	69	0	69,00	00:12,050	69	69	0	69,00	00:12,535	69	69	0	69,00	00:12,341
Pajek network Erdos collaboration network 971	7,00	7	7	0	7,00	00:00,215	7	7	0	7,00	00:00,202	7	7	0	7,00	00:00,203	7	7	0	7,00	00:00,204
Zachary's karate club	5,00	5	5	0	5,00	00:00,057	5	5	0	5,00	00:00,050	5	5	0	5,00	00:00,047	5	5	0	5,00	00:00,047
p_hat300-1	8,00	8	8	0	8,00	00:01,554	8	8	0	8,00	00:01,431	8	8	0	8,00	00:01,446	8	8	0	8,00	00:01,432
brock200_2	12,00	12	12	0	12,00	00:01,802	12	12	0	12,00	00:01,622	12	12	0	12,00	00:01,799	12	12	0	12,00	00:01,635
hamming8-4	16,00	16	16	0	16,00	00:03,837	16	16	0	16,00	00:03,403	16	16	0	16,00	00:03,266	16	16	0	16,00	00:03,429
brock200_4	17,00	17	15	0,67	16,30	00:02,962	17	15	0,70	16,40	00:02,635	17	15	0,82	16,30	00:02,549	17	15	0,67	16,30	00:02,599
brock400_2	29,00	24	22	0,70	22,40	00:10,555	25	23	0,70	23,40	00:09,203	23	22	0,52	22,40	00:09,195	23	22	0,52	22,40	00:09,375
brock400_4	33,00	25	22	0,82	23,00	00:11,310	33	23	4,03	25,40	00:09,567	23	22	0,48	22,30	00:09,639	25	22	0,97	22,60	00:09,484
gen200_p0.9_44	44,00	44	38	2,88	41,10	00:07,111	44	38	2,63	41,60	00:05,721	44	39	2,35	41,80	00:05,721	44	39	1,89	43,00	00:05,763
gen200_p0.9_55	55,00	55	55	0	55,00	00:07,823	55	55	0	55,00	00:05,774	55	55	0	55,00	00:05,883	55	55	0	55,00	00:06,042
gen400_p0.9_55	55,00	51	46	1,43	48,40	00:19,885	52	48	1,07	49,60	00:15,205	52	49	1,06	50,70	00:15,602	52	49	0,97	50,40	00:15,457
gen400_p0.9_65	65,00	63	47	4,85	51,20	00:20,296	65	47	6,41	54,00	00:15,920	65	52	4,16	63,30	00:15,653	65	59	1,90	64,40	00:15,475
gen400_p0.9_75	75,00	75	71	1,65	73,50	00:20,561	75	75	0	75,00	00:16,235	75	75	0	75,00	00:16,248	75	75	0	75,00	00:16,087
p_hat300-2	25,00	25	25	0	25,00	00:04,466	25	25	0	25,00	00:03,954	25	25	0	25,00	00:04,105	25	25	0	25,00	00:04,119
p_hat300-3	36,00	36	34	0,95	34,70	00:07,911	36	34	1,05	35,00	00:06,752	36	34	0,97	35,40	00:06,771	36	34	0,97	35,40	00:06,759
C250.9	44,00	44	41	1,03	42,20	00:10,093	44	42	0,82	43,00	00:07,561	44	41	0,95	42,70	00:07,921	44	42	0,70	43,60	00:08,082

C125.9	34,00	34	34	0	34,00	00:04,046	34	34	0	34,00	00:03,093	34	34	0	34,00	00:03,144	34	34	0	34,00	00:03,164
DSJC500_5	13,00	13	12	0,53	12,50	00:08,865	13	12	0,48	12,70	00:08,085	13	12	0,53	12,50	00:08,090	13	12	0,52	12,60	00:08,034
C500.9	57,00	54	49	1,43	51,60	00:28,402	54	50	1,42	51,30	00:21,910	55	49	1,75	52,80	00:23,216	55	47	3,03	51,50	00:22,668
brock800_2	29,00	20	18	0,63	18,80	00:27,341	20	18	0,47	19,00	00:27,400	19	18	0,32	18,10	00:27,277	19	18	0,52	18,60	00:27,360
brock800_4	33,00	19	18	0,52	18,40	00:28,054	20	18	0,67	19,30	00:27,008	19	18	0,52	18,60	00:26,219	20	18	0,67	18,70	00:26,305
keller5	27,00	27	24	0,92	25,20	00:33,436	27	25	0,82	25,70	00:29,722	25	24	0,48	24,70	00:30,002	27	24	0,88	25,10	00:30,076
p_hat700-1	11,00	11	9	0,67	9,70	00:07,802	11	9	0,79	10,20	00:07,019	11	9	0,67	9,70	00:07,039	10	9	0,48	9,70	00:07,138
p_hat700-2	44,00	43	42	0,42	42,80	00:18,218	44	42	0,70	43,40	00:16,113	44	44	0	44,00	00:16,635	44	44	0	44,00	00:16,382
p_hat700-3	62,00	61	59	0,82	60,30	00:31,241	62	59	1,03	60,80	00:29,231	62	61	0,32	61,90	00:29,083	62	61	0,32	61,90	00:29,178
MANN_a45	345,00	339	338	0,52	338,44	08:12,914	343	341	0,57	342,1	06:21,278	343	340	0,82	342,00	06:25,792	344	341	0,95	341,70	06:26,058
p_hat1500-2	65,00	63	60	0,95	61,70	01:11,268	65	62	1,03	63,2	01:12,092	65	64	0,48	64,70	01:12,239	65	65	0	65,00	01:12,950
p_hat1500-3	94,00	89	81	2,30	86,20	02:01,631	93	89	1,72	91,5	01:59,722	94	93	0,42	93,20	02:00,529	94	93	0,48	93,30	02:01,322
DSJC1000_5	15,00	14	13	0,48	13,30	00:30,734	14	13	0,52	13,6	00:31,037	14	13	0,48	13,30	00:31,519	14	13	0,53	13,50	00:31,960

Lampiran 3 Hasil Lengkap Pencarian *Best Known Maximum Clique* Dengan ACO, IACO, ACOPSO & IACOPSO

Dataset	*Best Known*	ACO							IACO						
		Max	*Max Iter*	*Avg Iter*	*Stdev Iter*	*Max Time*	*Avg Time*	*Stdev Time*	*Max*	*Max Iter*	*Avg Iter*	*Stdev Iter*	*Max Time*	*Avg Time*	*Stdev Time*
co-authorship of scientist in network theory and experiment	20.00	20	5	2.8	1,64	00:00,051	00:00.025	00:00,016	20	5	2.8	1,64	00:00,048	00:00.020	00:00,017
Dolphin network	5.00	5	2	1.2	0,45	00:00,017	00:00.011	00:00,005	5	1	1	0,00	00:00,005	00:00.003	00:00,002
email interchange network univ of rovia i virgili tarragon	12.00	12	15	5.2	5,63	00:00,136	00:00.051	00:00,052	12	15	5.2	5,63	00:00,128	00:00.042	00:00,050
Erdos991	7.00	7	12	5.4	5,18	00:00,086	00:00.043	00:00,037	7	12	6	4,90	00:00,087	00:00.041	00:00,035
facebook_combined	69.00	69	217	84.6	74,77	00:04,191	00:01.512	00:01,507	69	190	87	58,47	00:03,327	00:01.469	00:01,049
Pajek network Erdos collaboration network 971	7.00	7	24	10.8	9,20	00:00,212	00:00.082	00:00,078	7	17	6	6,56	00:00,090	00:00.036	00:00,036
Zachary's karate club	5.00	5	1	1	0,00	00:00,013	00:00.008	00:00,003	5	1	1	0,00	00:00,021	00:00.007	00:00,008
p_hat300-1	8.00	8	298	102.8	113,21	00:02,602	00:00.886	00:01,003	8	38	20.4	11,10	00:00,346	00:00.199	00:00,124
brock200_2	12.00	12	100	32	42,04	00:00,958	00:00.285	00:00,403	12	323	107.8	127,00	00:02,578	00:00.960	00:01,036
hamming8-4	16.00	16	46	39	8,72	00:00,418	00:00.347	00:00,082	16	41	23.8	15,71	00:00,392	00:00.220	00:00,154
brock200_4	17.00	17	3526	1122.8	1396,56	00:42,349	00:13.306	00:16,869	17	4630	1510.6	1880,36	00:59,914	00:19.035	00:24,444
brock400_2	29.00	29	1611	435.2	665,36	00:34,575	00:09.265	00:14,322	29	5191	1834.4	2065,56	02:04,482	00:43.038	00:49,798
brock400_4	33.00	33	5898	2885.4	2549,37	02:35,925	01:10.863	01:05,127	33	9320	5606.8	3443,23	03:36,631	02:07.523	01:19,104
gen200_p0.9_44	44.00	43	1352	313.8	580,41	00:24,020	00:05.484	00:10,363	44	67	52	13,13	00:01,230	00:00.919	00:00,259
gen200_p0.9_55	55.00	55	32	30.8	1,64	00:00,531	00:00.458	00:00,054	55	41	31	8,37	00:00,700	00:00.551	00:00,163
gen400_p0.9_55	55.00	50	93	68.6	15,52	00:02,288	00:01.512	00:00,457	50	139	91.2	29,24	00:03,527	00:02.232	00:00,821
gen400_p0.9_65	65.00	55	5369	1431	2291,78	02:16,430	00:40.269	00:58,697	59	3426	908.4	1450,52	01:37,663	00:27.971	00:41,357
gen400_p0.9_75	75.00	75	87	64.8	19,49	00:02,445	00:01.570	00:00,512	75	87	66.4	16,83	00:02,476	00:01.785	00:00,465
p_hat300-2	25.00	25	60	48.8	8,26	00:00,745	00:00.602	00:00,095	25	65	51.6	10,38	00:00,795	00:00.674	00:00,131
p_hat300-3	36.00	36	63	54.2	6,80	00:01,168	00:00.884	00:00,193	36	9861	2447.2	4242,72	03:15,096	00:48.241	01:23,987

C250.9	44.00	43	60	53.6	6,02	00:01,402	00:01.112	00:00,279	44	88	55.6	18,94	00:01,610	00:01.127	00:00,366
C125.9	34.00	34	34	29.4	4,34	00:00,320	00:00.291	00:00,033	34	34	28	5,15	00:00,514	00:00.341	00:00,102
DSJC500_5	13.00	13	13	743	627,92	00:39,946	00:17.392	00:14,657	13	3803	1884.4	1227,36	01:32,740	00:45.427	00:30,175
C500.9	57.00	52	50	1588.4	3318,38	04:55,929	01:02.056	02:10,746	51	95	79.4	14,88	00:03,558	00:02.783	00:00,699
brock800_2	29.00	20	19	2888.2	2337,48	05:17,044	02:15.673	01:50,994	24	4581	2107.8	1851,02	03:54,629	01:39.470	01:30,947
brock800_4	33.00	20	19	2590.2	2227,82	04:12,069	01:48.938	01:29,935	20	4743	1555.2	1866,43	04:00,304	01:13.903	01:36,139
keller5	27.00	27	26	4910.6	3623,21	06:59,815	03:48.435	02:50,496	27	8565	4788.8	3381,84	06:45,288	03:45.114	02:41,520
p_hat700-1	11.00	11	10	1433.2	994,38	01:00,902	00:33.437	00:25,470	11	3882	1743	1298,38	01:28,550	00:40.804	00:29,802
p_hat700-2	44.00	44	44	1097.4	1817,87	02:01,492	00:34.524	00:51,834	44	7753	1611.4	3433,27	05:01,303	01:01.975	02:13,790
p_hat700-3	62.00	62	60	72.8	6,22	00:04,566	00:03.482	00:00,950	61	8126	1683.6	3601,45	06:56,099	01:25.930	03:04,572
MANN_a45	345.00	340	339	1239.75	841,97	18:09,595	11:54.430	07:53,724	343	4583	2148	1850,19	45:05,697	20:10.719	18:14,441
p_hat1500-2	65.00	64	63	7216.4	1140,75	19:38,662	15:17.445	02:46,906	65	8719	7064	1998,23	19:19,347	15:33.180	04:48,631
p_hat1500-3	94.00	91	86	2317	4450,01	28:54,652	07:24.123	14:20,354	92	8660	2233.5	4284,34	23:23,085	06:00.964	11:34,748
DSJC1000_5	15.00	14	14	1367.75	941,92	01:44,971	00:55.288	00:40,741	14	9899	2922.25	4666,96	07:45,343	02:20.635	03:37,562

Dataset	Best Known	ACOPSO							IACOPSO						
		Max	Max Iter	Avg Iter	Stdev Iter	Max Time	Avg Time	Stdev Time	Max	Max Iter	Avg Iter	Stdev Iter	Max Time	Avg Time	Stdev Time
co-authorship of scientist in network theory and experiment	20.00	20	5	2.8	1,64	00:00,046	00:00.019	00:00,017	20	5	2.8	1,64	00:00,046	00:00.019	00:00,018
Dolphin network	5.00	5	1	1	0,00	00:00,003	00:00.001	00:00,001	5	1	1	0,00	00:00,006	00:00.003	00:00,003
email interchange network univ of rovia i virgili tarragon	12.00	12	15	5.2	5,63	00:00,097	00:00.034	00:00,036	12	15	5.2	5,63	00:00,115	00:00.037	00:00,044
Erdos991	7.00	7	12	7.4	3,44	00:00,067	00:00.044	00:00,018	7	12	5.6	4,72	00:00,067	00:00.033	00:00,028
facebook_combined	69.00	69	102	50	29,25	00:01,934	00:00.866	00:00,599	69	110	70.8	31,19	00:02,106	00:01.238	00:00,612
Pajek network Erdos collaboration network 971	7.00	7	53	16.8	20,98	00:00,359	00:00.108	00:00,143	7	22	14	6,67	00:00,146	00:00.091	00:00,047
Zachary's karate club	5.00	5	1	1	0,00	00:00,004	00:00.003	00:00,001	5	1	1	0,00	00:00,004	00:00.003	00:00,001
p_hat300-1	8.00	8	76	41.6	22,72	00:00,525	00:00.309	00:00,153	8	249	91.6	100,98	00:02,568	00:00.834	00:01,040
brock200_2	12.00	12	318	108.6	127,01	00:03,145	00:01.055	00:01,269	12	85	55.8	27,51	00:00,816	00:00.475	00:00,269
hamming8-4	16.00	16	57	36	21,62	00:00,652	00:00.344	00:00,239	16	62	31.8	20,54	00:00,782	00:00.364	00:00,289
brock200_4	17.00	17	3827	1940.4	1608,31	00:45,597	00:22.771	00:18,937	17	7622	3156.4	2656,68	01:32,717	00:38.727	00:32,345
brock400_2	29.00	24	8663	3908.6	3290,84	03:17,023	01:26.563	01:14,553	29	5437	3017.2	2312,58	02:09,143	01:08.439	00:52,984
brock400_4	33.00	33	2221	1118.2	704,38	00:49,797	00:24.114	00:16,271	33	6944	5028.6	2123,58	02:51,875	01:56.991	00:51,690
gen200_p0.9_44	44.00	44	9334	3213.8	4431,46	02:57,467	01:00.752	01:24,057	44	4783	1643.4	2084,00	01:35,144	00:32.102	00:41,157
gen200_p0.9_55	55.00	55	34	27.8	4,32	00:00,688	00:00.451	00:00,148	55	28	21.8	5,36	00:00,467	00:00.367	00:00,111
gen400_p0.9_55	55.00	51	4737	1762	2322,44	02:12,335	00:46.535	01:02,556	52	6632	1388	2931,53	02:55,349	00:36.676	01:17,522
gen400_p0.9_65	65.00	65	5470	1158	2410,52	03:25,672	00:43.310	01:30,764	65	7133	2876.8	3234,69	04:56,502	01:57.100	02:13,377
gen400_p0.9_75	75.00	75	5529	1151.8	2446,93	02:57,138	00:36.670	01:18,524	75	111	68.6	30,66	00:03,306	00:01.988	00:00,924
p_hat300-2	25.00	25	37	33.8	4,87	00:00,600	00:00.444	00:00,113	25	43	39.8	3,42	00:00,665	00:00.577	00:00,069
p_hat300-3	36.00	36	3787	1196.8	1597,43	01:12,358	00:22.745	00:30,580	36	89	69.2	19,36	00:01,687	00:01.257	00:00,384
C250.9	44.00	44	86	62.6	15,79	00:01,609	00:01.242	00:00,320	44	683	175.6	283,75	00:17,840	00:04.377	00:07,532
C125.9	34.00	34	28	23	3,81	00:00,282	00:00.226	00:00,047	34	40	26.6	9,91	00:00,390	00:00.284	00:00,070

DSJC500_5	13.00	13	3045	1287.8	1102,59	01:13,159	00:32.092	00:26,113	13	2513	1294.2	864,01	01:14,525	00:34.061	00:26,083
C500.9	57.00	55	9031	5255.6	3687,60	05:19,316	03:26.462	02:22,968	55	9220	6252.6	3306,80	05:42,836	04:03.455	02:06,567
brock800_2	29.00	19	1352	635	460,64	00:55,472	00:27.762	00:18,628	24	7838	3276.4	3078,87	07:21,438	03:14.026	03:05,704
brock800_4	33.00	20	9845	5891.6	3771,69	06:57,065	04:16.666	02:33,302	20	8441	3385.2	3268,53	05:58,850	02:30.845	02:15,351
keller5	27.00	27	9375	3527.8	3694,34	07:19,362	02:43.810	02:53,484	27	9568	6222.6	2265,63	07:27,527	04:51.893	01:45,046
p_hat700-1	11.00	11	4151	2569	1446,99	01:36,975	00:56.892	00:35,201	11	4762	2678.6	1854,89	01:59,130	01:02.645	00:47,688
p_hat700-2	44.00	44	121	81.2	25,00	00:03,208	00:02.382	00:00,896	44	2628	1035.4	1083,93	01:19,147	00:34.829	00:34,140
p_hat700-3	62.00	62	114	105.2	9,44	00:06,086	00:05.264	00:00,761	62	9675	2650.4	4163,36	09:04,113	02:34.634	03:56,168
MANN_a45	345.00	343	7216	2542.5	3162,72	44:31,902	32:41.226	48:07,870	344	4408	1957	1721,47	01:00,366	23:55.941	25:23,819
p_hat1500-2	65.00	65	1977	858.6	997,02	04:23,732	01:52.085	02:13,081	65	3150	1001.8	1236,76	07:08,340	02:14.057	02:49,480
p_hat1500-3	94.00	94	8832	5493.5	3446,97	28:56,386	16:33.742	10:58,950	94	9503	5250.5	3124,92	24:33,664	15:13.454	07:43,939
DSJC1000_5	15.00	14	9123	3573.5	4024,57	08:01,316	03:12.115	03:33,519	14	2684	1514.25	1098,82	02:36,897	01:28.154	01:05,117

DAFTAR RIWAYAT HIDUP

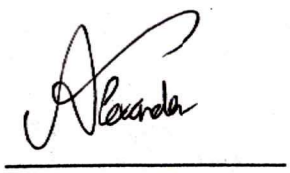
Nama	: Alexander Yap
Umur	: 22 tahun
Tempat / Tanggal Lahir	: Binjai / 17 Agustus 1994
Jenis Kelamin	: Pria
Agama	: Buddha
Tempat Tinggal	: Jl. Pajak Tavip No. 29 Blok B, Binjai

Pendidikan :

1. Tamatan SD Perguruan Kristen Methodist Indonesia-Binjai tahun 2006
2. Tamatan SMP Perguruan Kristen Methodist Indonesia-Binjai tahun 2009
3. Tamatan SMA Perguruan Kristen Methodist Indonesia-Binjai tahun 2012

Demikian daftar riwayat hidup ini saya perbuat dengan sesungguhnya.

Hormat Saya,



DAFTAR RIWAYAT HIDUP

Nama	: Jeffry Owen
Umur	: 22 tahun
Tempat / Tanggal Lahir	: Binjai / 31 Agustus 1994
Jenis Kelamin	: Pria
Agama	: Buddha
Tempat Tinggal	: Jl. Jendral Ahmad Yani no 316/12, Binjai

Pendidikan :

1. Tamatan SD Perguruan Kristen Methodist Indonesia-Binjai tahun 2006
2. Tamatan SMP Perguruan Kristen Methodist Indonesia-Binjai tahun 2009
3. Tamatan SMA Perguruan Kristen Methodist Indonesia-Binjai tahun 2012

Demikian daftar riwayat hidup ini saya perbuat dengan sesungguhnya.

Hormat Saya,