OPTIMASI *RESOURCE CONSTRAINED PROJECT SCHEDULING PROBLEM* (RCPSP) DENGAN ALGORITMA *SYMBIOTIC ORGANISM SEARCH* (SOS)

Yulius Candi[1], Andi[2] dan Doddy Prayogo[3]
[1] Mahasiswa Program Studi Magister Teknik Sipil, Universitas Kristen Petra, Surabaya.
[2, 3] Dosen Program Studi Magister Teknik Sipil, Universitas Kristen Petra, Surabaya.

[1] yuliuscandi@gmail.com, [2] andi@petra.ac.id, [3] prayogo@petra.ac.id

ABSTRAK: Proyek konstruksi yang tidak memiliki *resource* yang cukup dapat mengakibatkan proyek terlambat. Permasalahan ini disebut juga *resource constrained project scheduling problem* (RCPSP) Penelitian ini menggunakan algoritma metaheuristik untuk menyelesaikannya. *Symbiotic organism search* (SOS) merupakan algoritma metaheuristik yang dibuat oleh Cheng dan Prayogo dapat menyelesaikan bermacam-macam permasalahan numerik yang rumit. Algoritma ini mempunyai tiga fase utama, yaitu mutualisme, komensalisme, dan parasitisme. Penelitian ini dilakukan menggunakan jadwal proyek yang sedang berjalan di Surabaya, dengan membatasi *resource* sebanyak 75%, 50%, dan kebutuhan paling minimal dari total *resource* mula-mula. Dari hasil percobaan, algoritma SOS terbukti dapat menyelesaikan RCPSP, namun percobaan pembatasan *resource* paling minimal menghasilkan *total duration* yang lebih lambat dari hasil program pembanding. Algoritma SOS membutuhkan banyak iterasi dan langsung mendapatkan hasil optimal, sedangkan *solver* program pembanding membutuhkan proses yang berulang hingga mendapatkan hasil yang optimal.

Kata kunci: *resource constrained project scheduling problem, optimasi, metaheuristik, symbiotic organism search*

ABSTRACT: *The construction project which has insufficient resources may cause delay. This is known as resource constrained project scheduling problem (RCPSP). This research used a metaheuristic algorithm to solve it. Symbiotic organisms search (SOS), which was invented by Cheng and Prayogo, is a metaheuristic algorithm that is able to solve various complex numerical problems. This algorithm has three main phases, namely, mutualism, commensalism, and parasitism. This research used a schedule of the ongoing project in Surabaya and limited its resources to 75%, 50%, and the minimum requirement from its original amount. The results confirmed the excellent performance of SOS in solving RCPSP, however the total duration from the minimum requirement resource limitation is much later compared with other selected program. The SOS algorithm requires many iterations, but directly obtains the optimal result. Other program requires repeated process until the optimal result is obtained.*

Keywords: *resource constrained project scheduling problem, optimization, metaheuristic, symbiotic organism search*

1. PENDAHULUAN

Proyek membutuhkan *resource* untuk membangunnya. Ketika *resource* yang dibutuhkan tidak cukup, proyek dapat terlambat dari jadwal yang sudah direncanakan. *Resource* yang tidak cukup akan membuat pekerjaan-pekerjaan harus bergeser mundur. Memastikan tersedianya sumber daya yang cukup untuk menangani proyek-proyek yang sedang berjalan membuat penjadwalan semakin rumit (El-abbasy et al., 2016). Masalah ini disebut sebagai *resource constrained project scheduling problem* (RCPSP) (Zhang & Tam, 2006).

Permasalahan ini dapat diselesaikan dengan mengoptimasi pemberdayaan *resource*. Tujuan dari penyelesaian ini adalah untuk meminimalkan total durasi proyek dengan mengatur penjadwalan dari aktivitas suatu proyek sedemikian rupa sehingga pengutamaan antara aktivitas dan keterbatasan *resource* dapat terpenuhi (Koulinas et al., 2014). Hal ini dapat dilakukan dengan cara menggeser jadwal dari aktivitas yang mempunyai *float* atau *slack* lebih banyak, sehingga ketersediaan *resource* dapat diatur. Perumusan untuk penyelesaian RCPSP ini ditemukan oleh Talbot dan Patterson (Zhang & Tam, 2006).

RCPSP merupakan masalah optimasi, sehingga untuk memudahkan penyelesaian dari RCPSP ini digunakan algoritma metaheuristik. Algoritma metaheuristik merupakan algoritma yang dapat menyelesaikan masalah optimasi yang kompleks. Dalam perkembangannya, algoritma metaheuristik terinspirasi dari alam dan menghasilkan algoritma-algoritma baru, beberapa di antaranya adalah *migrating birds optimization* (MBO), *particle swarm optimization* (PSO), *cuckoo search algorithm* (CSA), dan lain sebagainya (Desale et al., 2015). Hampir semua algoritma metaheuristik terbaru mempunyai karakteristik yang sama, yaitu terinspirasi dari alam, memakai variabel acak, tidak memerlukan informasi gradien yang substansial, dan mempunyai beberapa parameter yang dapat dipakai untuk masalah tertentu (Cheng & Prayogo, 2014).

Salah satu algoritma metaheuristik baru yang mampu menyelesaikan permasalahan optimasi yang kompleks dalam berbagai bidang adalah *symbiotic organism search* (SOS). Algoritma ini dibuat oleh Cheng dan Prayogo (2014). Algoritma ini bersifat kontinu dan terdiri dari tiga fase utama, yaitu mutualisme, komensalisme, dan parasitisme. Kelebihan dari algoritma ini adalah tidak memerlukan parameter yang spesifik (Cheng & Prayogo, 2014). Algoritma ini juga telah digunakan untuk menyelesaikan beberapa masalah dan terbukti mengungguli algoritma lainnya seperti *genetic algorithm* (GA), *differential evolution* (DE), dan *particle swarm optimization* (PSO) (Cheng et al., 2016).

Sampai saat ini belum ada penyelesaian RCPSP dengan menggunakan algoritma SOS. Dengan memakai algoritma SOS, penyelesaian terhadap permasalahan penanganan sumber daya dapat dioptimalkan. Data penelitian ini menggunakan jadwal proyek nyata yang sedang berjalan ketika penelitian ini dilakukan. Pada penelitian ini perlu memodifikasi algoritma SOS yang awalnya dibuat untuk menyelesaikan masalah optimasi yang bersifat kontinu menjadi diskrit. Untuk perbandingan hasil dengan algoritma SOS ini digunakan *solver* dari program *Microsoft Project* 2010.

2. TINJAUAN PUSTAKA

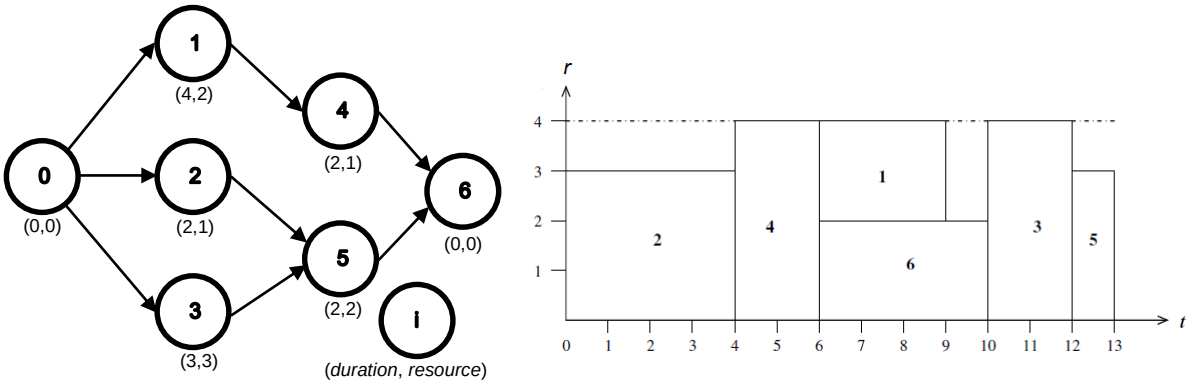
2.1 *Resource Constrained Project Scheduling Problem* (RCPSP)

Resource constrained project scheduling problem (RCPSP) merupakan proses penyelesaian penjadwalan dengan mempertimbangkan antara keterbatasan *resource* dan keterbatasan pengutamaan akivitas (Zhang et al., 2005). Tujuan utama dari RCPSP adalah untuk meminimalkan total durasi proyek dengan mengatur penjadwalan dari aktivitas dalam suatu proyek sedemikian rupa sehingga pengutamaan dalam antara aktivitas dan keterbatasan *resource* dapat terpenuhi (Koulinas et al., 2014).

RCPSP dalam penelitian berdasarkan asumsi-asumsi berikut: (1) aktivitas-aktivitas yang tersusun dalam proyek memiliki jangka waktu tertentu dan diketahui; (2) semua *predecessor* harus selesai sebelum suatu kegiatan dapat dijalankan (*precedence constraints*); (3) *resource* bisa saja bervariasi yang tersedia dalam jumlah terbatas dan dapat diperbarui dari waktu ke waktu (*multiple resource constraints*); (4) kegiatan yang didahulukan tidak dapat diganggu pada saat sedang berjalan; (5) tujuan manajerial adalah untuk meminimalkan durasi proyek (Zhang & Tam, 2006). Data proyek untuk penyelesaian RCPSP ini berupa *network* aktivitas dan data *resource* (Vanchouke, 2018).

Network merupakan relasi antara aktivitas-aktivitas yang dimulai dan penerus dalam proyek. Relasi *precedence* adalah relasi *finish-to-start* minimum dengan jeda waktu nol, dan tidak ada perpanjangan untuk relasi *start-to-start*, *finish-to-finish*, atau *start-to-finish*, juga tidak adanya waktu maksimum keterlambatan yang diperhitungkan. *Resource* merupakan sumber daya yang dipakai dalam sebuah proyek, baik material, alat-alat pertukangan, ala-alat berat, para pekerja, dan lain sebagainya (Vanchouke, 2018). *Resource* dalam penelitian ini menggunakan sumber daya manusia.

Data proyek berisikan pekerjaan-pekerjaan yang di dalamnya terdapat *predecessor* dari pekerjaan satu sama lain, durasi pekerjaan, dan jumlah *resource*. Kemudian data tersebut dimodelkan ke dalam *network* pekerjaan. Network akan memudahkan memasukkan data pekerja ke dalam *resource diagram*. Contoh *network diagram* dan *resource diagram* dapat dilihat pada Gambar 1.



Gambar 1. *Network Diagram* dan *Resource Diagram*

Formulasi dari RCPSP pertama kali ditemukan oleh Talbot dan Patterson sebagai berikut:

$$\min\{max f_i | i = 1,2,\dots,N\} \quad (1)$$

$$f_j - f_i \geq d_i \quad \forall j \in P_i; \quad i = 1,2,\dots,N \quad (2)$$

$$\sum_{A_t} r_{ik} \leq R_k, \quad \mathrm{k} = 1,2,\dots,K; \quad t = S_i, S_2, \dots, S_N \quad (3)$$

N adalah jumlah dari aktivitas yang terlibat dalam proyek, $\boldsymbol{f_i}$ adalah waktu *finish* dari aktivitas *i* (*i*=1,...,*N*); $\boldsymbol{d_i}$ dalah durasi dari aktivtias *i*, $\boldsymbol{P_i}$ adalah sekumpulan dari aktivitas-aktivitas yang sudah dijadwalkan (*predecessor*) sebelum aktivitas *i* dapat dikerjakan; $\boldsymbol{R_k}$ adalah jumlah *resource k* yang tersedia (*k*=1,...,k) dan *k* adalah jumlah dari macam *resource*; $\boldsymbol{r_{ik}}$ adalah jumlah *resource* yang dibutuhkan oleh aktivitas *i*, dan a_t sekumpulan dari aktivitas yang sedang berjalan pada waktu *t* dan $\boldsymbol{s_i}$ (=$\boldsymbol{f_i} - \boldsymbol{d_i}$) adalah waktu *start* dari aktivitas *i*. persamaan (1) sebagai *objective*, sementara persamaan (2) dan (3) sebagai *precedence constraints* dan *resource contraints* (Zhang & Tam, 2006).

2.2 Algoritma Metaheuristik

Metaheuristik secara formal didefinisikan sebagai proses generasi yang iteratif yang memandu subordinat heuristik dengan menggabungkan konsep yang berbeda secara cerdas untuk mencari dan memanfaatkan ruang pencarian, belajar strategi yang digunakan untuk menyusun informasi agar menemukan solusi yang mendekati optimal secara efisien. Berikut sifat-sifat yang merupakan ciri dari sebagian besar metaheuristik (Desale et al., 2015):

a. Merupakan strategi yang memandu proses pencarian. Tujuannya adalah untuk mencari secara efisien ruang pencarian untuk mendapatkan solusi yang mendekati optimal.
b. Merupakan teknik algortima metaheuristik mulai dari prosedur pencarian lokal sederhana sampai proses pembelajaran yang kompleks.
c. Algoritma metaheuristik adalah perkiraan dan biasanya hasil percarian tidak akan sama.

2.3 *Symbiotic Organism Search* (SOS)

Symbiotic Organism Search (SOS) pertama kali ditemukan oleh Cheng dan Prayogo merupakan algoritma yang mensimulasikan perilaku interaksi antara organisme di alam. Organisme sangat jarang hidup sendiri karena membutuhkan bantuan dari spesies lain untuk makan dan bahkan bertahan hidup. Hubungan kebergantungan inilah yang disebut sebagai simbiosis (Cheng & Prayogo, 2014).

Hubungan simbiosis yang paling sering ditemukan di alam adalah simbiosis mutualisme, komensalisme, dan parasitisme. Mutualisme menunjukan sebuah hubungan simbiosis antara dua spesies berbeda yang saling menguntungkan. Komensalisme adalah hubungan simbiosis antara dua spesies berbeda yang satunya diuntungkan dan yang lainnya tidak terpengaruh atau netral. Parasitisme adalah hubungan simbiosis antara dua spesies berbeda yang satunya diuntungkan dan lainnya secara aktif dirugikan. Usulan algoritma ini dikembangkan untuk menyelesaikan optimasi numerik pada ruang percarian secara kontinu.

SOS dimulai dengan inisiasi populasi disebut sebagai ekostistem. Dalam inisiasi ekosistem, kelompok organisme dibuat secara acak pada ruang pencarian. Setiap organisme dalam ekosistem saling terkait dengan nilai *fitness* tertentu yang menggambambarkan tingkat adaptasi dari tujuan yang diinginkan (Cheng & Prayogo, 2014). Setiap organisme berinteraksi dengan organisme lainnya secara acak di setiap fase. Prosesnya terus diulang sampai

termination criteria sudah memenuhi. Berikut gambaran urutan algoritma dari fase di atas: *Initialization*, *Repeat* (fase mutualisme, komensalisme, parasitisme), *Until* (*termination criterion* telah memenuhi).

a. Fase Mutualisme

Pada fase ini, X_i adalah organisme yang dicocokkan ke anggota ke-*i* dari sebuah ekosistem. Kemudian, organisme yang lain X_j dipilih secara acak dari sebuah ekosistem untuk berinteraksi dengan X_i. Masing-masing organisme terjalin dalam hubungan mutualisme yang bertujuan untuk meningkatkan *survival advantage* mereka dalam ekosistem. X_i dan X_j tersebut dihitung untuk menjadi kandidat solusi baru, yang dirumuskan menjadi Persamaan (4) dan (5).

$X_{inew} = X_i + \text{rand}(0,1) \times (X_{best} - Mutual\_Vector \times BF_1)$ (4)

$X_{jnew} = X_j + \text{rand}(0,1) \times (X_{best} - Mutual\_Vector \times BF_2)$ (5)

$Mutual\_Vector = \frac{X_i + X_j}{2}$ (6)

rand(0,1) pada persamaan (4) dan (5) adalah vektor dari angka acak. BF1 dan BF2 adalah *benefit factor* yang ditentukan acak antara 1 atau 2. Nilai ini merupakan tingkat keuntungan untuk setiap organisme. Persamaan (6) menunjukkan vektor yang bernama "*Mutual_Vector*" yang mewakili relasi dari organisme X_i *dan* X_j. Bagian dari persamaan, (X_{best} - *Mutual_Vector* * BF_1), menggambarkan usaha saling menguntungkan untuk mencapai tujuan mereka dalam meningkatkan *survival advantage* mereka. X_{best} dibutuhkan karena mewakili sebagai tingkat beradaptasi yang paling tinggi. Selain itu, X_{best} / solusi global dipakai untuk memodelkan tingkat adaptasi tertinggi sebagai poin target untuk *fitness* untuk masing-masing organisme. Pada akhirnya organisme akan diperbarui hanya ketika *fitness* baru mereka lebih baik dibanding sebelumnya.

b. Fase Komensalisme

Sama dengan fase mutualisme, organisme X_j dipilih secara acak dari sebuah ekosistem untuk berinteraksi dengan X_i. Dalam keadaan ini, organisme X_i berupaya untuk mendapat keuntungan dari interaksi tersebut. Namun, organisme X_j sendiri tidak mendapat keuntungan atau dirugikan dari hubungan tersebut. Kandidat solusi baru X_i dihitung berdasarkan dari simbiosis komensalisme antara organisme X_i dan X_j, yang dimodelkan dalam persamaan (7). Dengan tetap mengikuti aturan, organisme X_i akan diperbarui hanya ketika *fitness* barunya lebih baik dari sebelumnya.

$X_{inew} = X_i + \text{rand}(-1,1) \times (X_{best} - X_j)$ (7)

Bagian dari persamaan, (X_{best} - X_j), menggambarkan keuntungan yang *beneficial advantage* dari X_j untuk menolong X_i untuk meningkatkan *survival advantage* dalam ekosistem sampai ke tingkat tertinggi dari organisme sekarang (diwakilkan dari X_{best}).

c. Fase Parasitisme

Dalam fase ini, organisme X_i diberikan sebuah peran dengan menciptakan sebuah parasit buatan yang disebut "*Parasite_Vector*". *Parasite_Vector* ini dibuat dalam ruang pencarian dengan menduplikat organisme X_i, lalu memodifikasi dimensi yang dipilih secara acak menggunakan angka acak. Organisme X_j dipilih secara acak dari sebuah ekosistem dan melayani sebagai wadah dari vektor parasit. *Parasite_Vector* mencoba menggantikan X_j dalam ekosistem. Masing-masing organisme kemudian dievaluasi untuk mengukur *fitness* mereka. Jika *Parasite_Vector* memiliki nilai *fitness* yang lebih baik, ia akan membunuh

organisme X_j dan mengambil posisinya dalam ekosistem. Jika nilai *fitness* dari X_j lebih baik, X_j akan kebal dari parasit dan *Parasite_Vector* tidak akan dapat hidup di dalam ekosistem itu lagi.

Setiap fase di atas tetap mengambil *fitness* terbaik dari masing-masing organisme yang akan menghasilkan solusi yang optimal. Fase-fase di atas dapat dilihat pada *flowchart* pada Gambar 2.

3. METODOLOGI PENELITIAN

3.1 Diagram Alur Metodologi Penelitian

Penelitian ini diawali dengan melakukan studi literatur dari jurnal-jurnal mengenai *resource constrained* dan algoritma metaheuristik yang dipakai dalam penelitian ini adalah SOS. Kemudian data dikumpulkan berupa jadwal dan data jumlah *resource* proyek yang diteliti. Data-data yang terkumpul kemudian dibuat permodelan untuk dapat dimasukkan ke dalam algoritma metaheuristik. Lalu data tersebut diolah sedemikian rupa sampai mendapatkan hasil yang optimal. Algoritma metaheuristik ini memakai program MATLAB 2017a. Hasil dari olahan data tersebut akan dimasukan dalam bentuk laporan. Kerangka penelitian dapat dilihat pada Gambar 3.

3.2 Studi Literatur

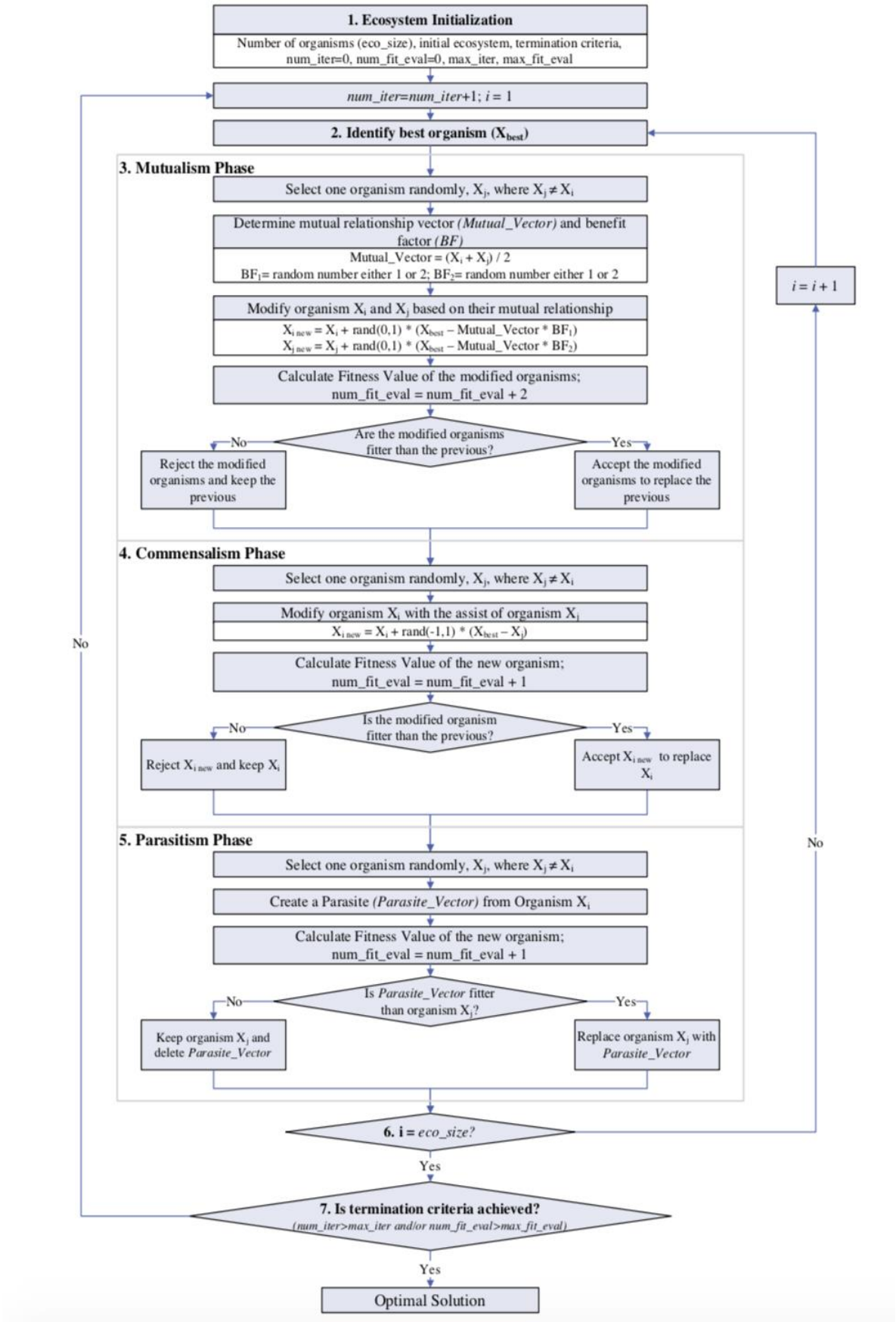
Studi literatur dilakukan dengan meneliti jurnal-jurnal dan penelitian-penelitian yang membahas tentang *resource constrained*. Dilanjutkan dengan studi mengenai algoritma yang akan digunakan dalam penelitian ini, yaitu SOS.

3.3 Pengumpulan Data

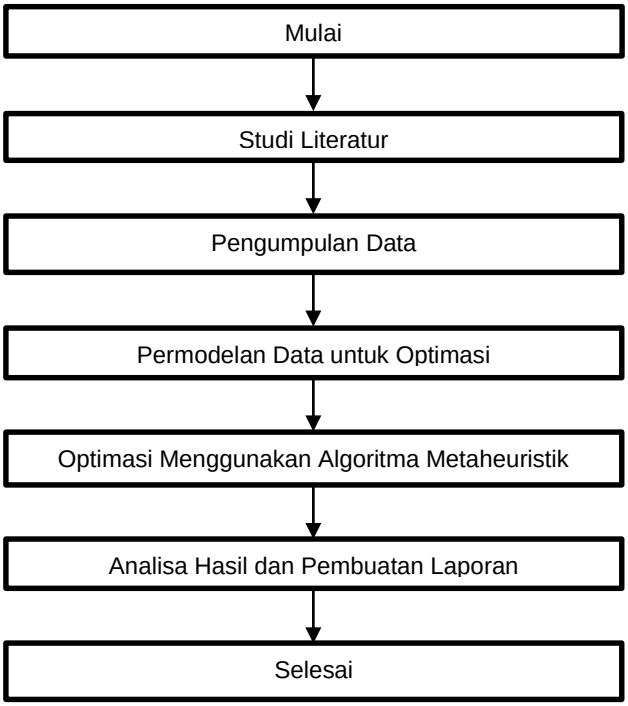
Data yang dikumpulkan untuk penelitian ini berupa jadwal proyek nyata dan data jumlah pekerja yang diperlukan pada setiap aktivitas. Data-data yang diperlukan untuk penelitian ini adalah identitas aktivitas, jumlah *resources* yang dibutuhkan, durasi setiap aktivitas, *predecessor* masing-masing aktivitas, dan *start time* setiap aktivitas. Data-data ini dimodelkan dan dimasukkan ke dalam program untuk diolah dalam formula algoritma SOS.

3.4 Permodelan Data untuk Optimasi

Permodelan dilakukan dengan mengolah data yang sudah terkumpul yang telah melalui pengecekan terhadap *predecessor* masing-masing aktivitas. Data tersebut kemudian dimodelkan ke dalam *array*/matriks berisikan, jumlah pekerja, dan durasi dari masing-masing aktivitas, serta dimasukan jumlah *resource constraint* untuk pengolahan data tersebut.



Gambar 2. Alur Kerja Proses Algoritma SOS
Sumber: Cheng & Prayogo, 2014



Gambar 3. Alur Kerangka Penelitian

Permodelan dilakukan dengan cara berikut:

a. Permodelan untuk *Resources*
   $res = [r_1 , r_2 , r_3 , \dots , r_n]$
b. Permodelan untuk *Predecessor*
   $pre = [p1_1 \dots p1_m ; p2_1 \dots p2_m ; p3_1 \dots p3_m ; \dots ; pn_m]$
c. Permodelan untuk Durasi
   $dur=[d_1,d_2,d_3,\dots,d_n]$

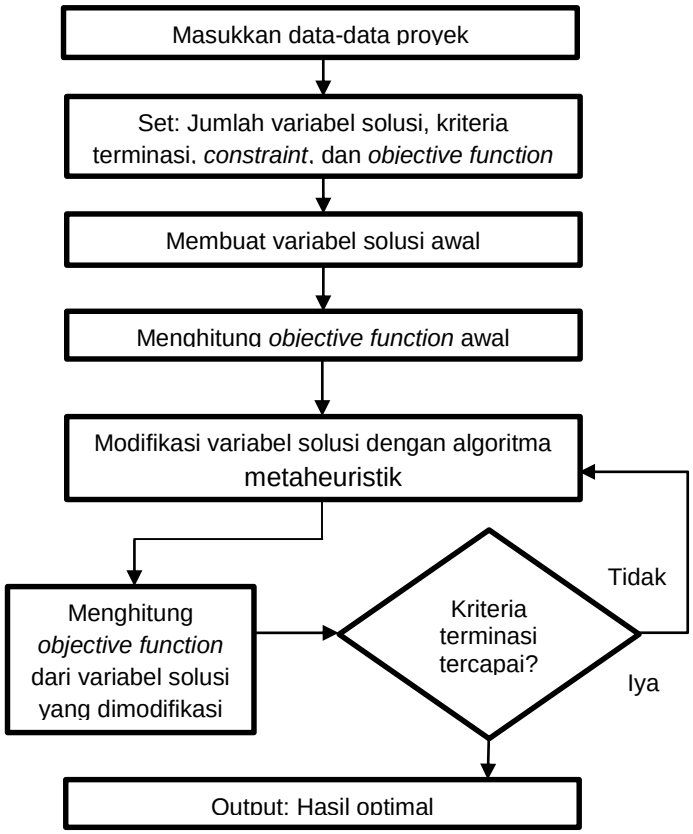
n merupakan urutan item pekerjaan yang dimasukan, *res* merupakan kebutuhan *resources* untuk aktivitas *n*, pn_m adalah *Predecessor* ke-*m* dari aktivitas *n*, *dur* adalah durasi dari aktivitas *n*.

3.5 Optimasi Menggunakan Algortima Metaheuristik

Data yang sudah dimodelkan ke dalam matriks akan diproses untuk dioptimasi menggunakan algoritma SOS dengan menggunakan program MATLAB 2017a. Percobaan pengoptimasian dilakukan dengan membatasi *resources* dari pekerjaan struktur dan arsitektur sebesar 75%, 50%, dan 25% dari total ketersediaannya masing-masing. Alur proses optimasi dengan menggunakan algoritma metaheuristik dapat dilihat pada Gambar 4.

Sebelum proses optimasi dilakukan, data-data yang didapatkan dimasukkan ke dalam program dengan permodelan seperti penjelasan di atas. Dilanjutkan dengan proses optimasi dengan menentukan jumlah variabel, kriteria terminasi (iterasi), *constraint* dan *cbjective function*. Iterasi merupakan proses pengulangan dalam program dalam mengolah data agar pengolahan data semakin dekat dengan hasil yang optimal. Variabel solusi adalah data yang diolah berisikan set *start time* setiap aktivitas. Jumlah variabel solusi dan iterasi tidak memiliki formula yang tepat untuk mendapatkannya. Dalam penelitian ini jumlah variabel solusi dan iterasi ditentukan dari jumlah yang kecil lalu ditambah sampai mendapatkan hasil yang paling

optimal yang bisa dicapai atau konvergen. Percobaan jumlah iterasi ditentukan dari 100 dan terus ditambah dengan kelipatan 100 sampai hasil optimasi konvergen. Jumlah variabel solusi ditentukan tetap pada angka 100.



Gambar 4. Alur Optimasi Menggunakan Algoritma Metaheuristik

Setelah semua pengaturan di awal sudah ditentukan, program membuat variabel solusi awal dengan membuat set *start time* untuk semua aktivitas sebanyak jumlah variabel solusi yang ditentukan untuk diolah dalam program. Kemudian semua variabel solusi awal dicek berdasarkan *constraints* dan o*bjective function* yang sudah ditentukan di awal untuk mencari variabel solusi terbaik. Variabel solusi terbaik diukur dengan nilai *objective function*. Nilai *objective function* adalah *total duration* yang didapatkan pada set *start time* dalam setiap variabel solusi. Pencarian variabel solusi terbaik sudah menjadi aturan dari algoritma metaheuristik untuk dijadikan sebagai patokan dalam proses pengoptimasian.

Variabel solusi tersebut kemudian diolah dalam proses algoritma SOS dengan membuat *start time* dalam variabel solusi bergeser berdasarkan formulasi modifikasi dari algoritma SOS. Variabel solusi yang bergeser dibatasi dengan adanya *lower bound* dan u*pper bound* yang sudah ditentukan sebelum percobaan. Batasan ini digunakan untuk mengurangi probabilitas jawaban yang terlalu banyak, sehingga pencarian dapat lebih cepat, akurat, dan tidak menghasilkan banyak *slack* sehingga membuat t*otal duration* semakin panjang. *Lower bound* yang ditentukan menggunakan e*arly start* agar aktivitas tidak bergeser sebelum *predecessor* aktivitas tersebut. *Upper bound* yang ditentukan adalah total dari semua durasi aktivitas karena *total duration* tidak dapat melebihi dari jumlah semua durasi aktivitas.

Variabel solusi yang sudah melalui proses modifikasi dari algoritma SOS dicek kembali berdasarkan *constraints* dan *objective function* untuk mencari variabel solusi terbaik. Variabel solusi terbaik terus diperbarui sampai kriteria terminasi tercapai. Setelah kriteria terminasi tercapai, variabel solusi terbaik yang terus diperbarui adalah hasil yang optimal, yaitu durasi terpendek berdasarkan o*bejctive function* tanpa melanggar *constraints.*

3.6 Analisis Hasil dan Pembuatan Laporan

Hasil yang paling optimal dari algoritma metaheuristik SOS akan dicatat. Kemudian hasil dari algoritma tersebut akan dibandingkan dengan *solver* dari program *Microsoft Project* 2010 untuk melihat hasil yang paling optimal.

4. HASIL DAN PEMBAHASAN

4.1 Pengumpulan Data

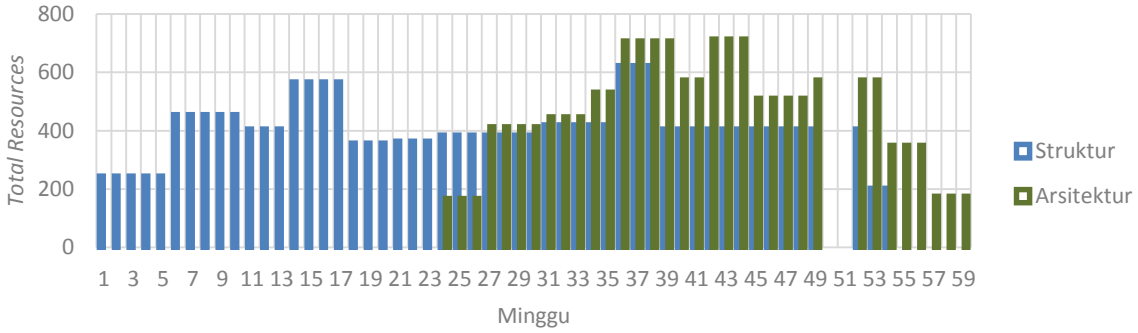
Data jadwal proyek yang digunakan adalah proyek konstruksi gereja yang berlokasi di Surabaya yang sedang berjalan selama penelitian ini dilakukan. Data yang dikumpulkan berupa *schedule* pekerjaan dan jumlah *resource* mingguan. Data-data yang diperoleh dari jadwal proyek dapat dilihat pada Tabel 1.

Tabel 1. Data Jadwal Proyek X

NO	ITEM PEKERJAAN	PEKERJA (org minggu)	PREDECESSOR	DURASI (minggu)	START TIME (minggu ke)
A	**PEKERJAAN STRUKTUR**				
1	Lantai 1	245	0	10	1
2	Lantai 2	210	1SS+5	12	6
3	Lantai 2M	161	2SS+8	7	14
4	Lantai 2A	196	2SS+5	13	11
5	Lantai 2AM	168	3FS,4SS+10	10	21
6	Lantai 3	217	4FS,5SS+3	15	24
7	Lantai 4	203	6SS+7	10	31
8	Lantai 5	203	7SS+5	10	36
9	Lantai 6	203	8SS+5	10	41
10	Office Lantai Atap	203	9SS+5	7	46
B	**PEKERJAAN ARSITEKTUR**				
11	Lantai Lower Ground	168	6SS	7	24
12	Lantai 1	245	1FS,11SS+3	13	27
13	Lantai 2	203	2FS,12SS+4	14	31
14	Lantai 2M	84	3FS,13SS+3	8	34
15	Lantai 2A	175	4FS,13SS+5	10	36
16	Lantai 2AM	112	5FS,15SS+4	9	40
17	Lantai 3	224	6FS,16SS+2	10	42
18	Lantai 4	175	7FS,17SS+4	6	46
19	Lantai 5	175	8FS,18SS+3	6	49
20	Lantai 6	175	9FS,19SS+3	6	54

Data dari Tabel 1 dimodelkan dalam bentuk *resource diagram* yang dapat dilihat pada Gambar 5. Berdasarkan Gambar 5, maka didapatkan jumlah pekerja maksimal untuk pekerjaan struktur adalah 623 orang/minggu dan pekerjaan arsitektur adalah 714 orang/minggu. Jumlah *resources* tersebut dianggap menjadi jumlah *resource* yang tersedia. Jumlah *resources* ini

nantinya pada saat program dijalankan untuk melihat kemampuan program dalam menyelesaikan RCPSP.



Gambar 5. *Resource Diagram* Proyek X

4.2 Permodelan Data untuk Optimasi

Data yang dibutuhkan untuk dianalisa terdiri dari identitas aktivitas, *predecessor*, durasi, dan jumlah *resources*. Semua informasi yang dibutuhkan pada Tabel 1 dimasukkan ke dalam program dalam bentuk *arrray*/matriks. Dalam jadwal proyek terdapat aktivitas dengan hubungan *Start-to-Start*, maka dari itu perlu menambah matriks untuk memasukkan informasi dari hubungan tersebut dengan dimodelkan menjadi $ss = [\, ss1_1 \ldots ss1_m \,;\, ss2_2 \ldots ss2_m \,;\, ss3_1 \ldots ss3_m \,;\, \ldots \,;\, ssn_m \,]$. ssn_m merupakan *lag* dari hubungan *precedential Start-to-Start* dari *Predecessor* ke *m* dari aktivitas *n.* Agar memudahkan pembuatan program, maka durasi hubungan *Finish-to-Start* dikonversi menjadi durasi *lag* pada hubungan *Start-to-Start*. Caranya dengan menggantinya dengan durasi aktivitas sebelumnya. Sebagai contoh aktivitas 5 pekerjaan struktur Lantai 2AM mempunyai hubungan *Finish-to-Start* dengan aktivitas 3. Maka pada permodelan durasi *lag Start-to-Start* diisi dengan durasi dari aktivitas 3, yaitu 7 minggu.

Berikut permodelan untuk semua data-data yang diperoleh:

a. Permodelan untuk *Resources*
```
res=[35;30;23;28;24;31;29;29;29;29;24;35;29;12;25;16;32;25;25;25]
```

b. Permodelan untuk *Predecessor*
```
pre=[0 0;1 0;2 0;2 0;3 4;4 5;6 0;7 0;8 0;9 0;6 0;1 11;2 12;3 13;4
13;5 15;6 16;7 17;8 18;9 19]
```

c. Permodelan untuk Durasi hubungan *Start-to-Start*
```
ss=[0 0;5 0;8 0;5 0;7 10;13 3;7 0;5 0;5 0;5 0;0 0;10 3;12 4;7 3;13
5;10 4;15 2;10 4;10 3;7 3]
```

d. Permodelan untuk Durasi
```
dur=[10;12;7;13;10;15;10;10;10;7;7;13;14;8;10;9;10;6;6;6]
```

4.3 Pengoptimasian Menggunakan SOS

Variabel yang diukur adalah *Start Time* masing-masing aktivitas dengan total 20 variabel (aktivitas). Program dimulai dengan *generate Start Time* secara acak untuk setiap aktivitas,

dan membuat satu variabel solusi. Variabel solusi ini yang dimodifikasi oleh program untuk mendapatkan hasil yang optimal.

Sebelum memulai percobaan, program dimulai dengan mencari *Start Time* tanpa pembatasan *resource*. Dengan hasil yang sama seperti *early start* dari jadwal proyek, maka program dapat dipakai untuk menyelesaikan pembatasan *resource.* Percobaan dalam penelitian ini dilakukan dengan membatasi *resources* (R_k) dari pekerjaan struktur dan arsitektur sebanyak 75%, 50%, dan 25% dari total ketersediaannya masing-masing. Percobaan pertama dilakukan dengan membatasi kebutuhan *resource* sebanyak 75% pada pekerjaan struktur yang semula mempunyai total 623 orang/minggu menjadi 468 orang/minggu, dan untuk pekerjaan arsitektur yang semula mempunyai total 714 orang/minggu menjadi 536 orang/minggu. Percobaan kedua dilakukan dengan membatasi kebutuhan *resource* sebanyak 50% untuk pekerjaan struktur menjadi 312 orang/minggu, dan untuk pekerjaan arsitektur menjadi 357 orang/minggu. Percobaan ketiga dilakukan dengan membatasi kebutuhan *resource* sebanyak 25% untuk pekerjaan struktur menjadi 156 orang/minggu, dan untuk pekerjaan arsitektur 179 orang/minggu.

Keterbatasan 25% pada *resource* dari pekerjaan struktur dan pekerjaan arsitektur tidak dapat diaplikasikan karena tidak dapat memenuhi kebutuhan dari aktivitas yang membutuhkan *resource* lebih dari keterbatasan itu. Maka dari itu, dipilih aktivitas yang memiliki kebutuhan *resource* yang paling banyak, sehingga ditentukan kebutuhan *resources* untuk pekerjaan struktur ditentukan sebanyak 245 orang/minggu berdasarkan Akitvitas 1 dan untuk pekerjaan arsitektur ditentukan sebanyak 245 orang/minggu berdasarkan Aktivitas 12. Dengan begitu, kebutuhan *resource* dari semua aktivitas dapat terpenuhi.

Dalam percobaan ini ditentukan jumlah iterasi dan variabel solusi dari jumlah kecil sampai besar untuk mengetahui kemampuan program. Jumlah iterasi dicoba dari 100, diteruskan berkelipatan 100 sampai seterusnya. Jika dalam percobaan penambahan iterasi terakhir tidak ada perubahan nilai sebanyak 100 kali, percobaan dengan penambahan iterasi dihentikan karena hasil sudah konvergen. Jumlah variabel solusi ditentukan tetap pada angka 100.

Berdasarkan data *predecessor* di Tabel 1, dari jadwal proyek yang didapatkan terdapat hubungan *Start-to-Start*. Adanya hubungan ini membuat *constraint* 1 di Persamaan (2) tidak dapat dilakukan, karena Persamaan (2) hanya untuk aktivitas dengan hubungan *Finish-to-Start*, tidak dapat digunakan pada hubungan *Start-to-Start*. Modifikasi Persamaan (2) ini tetap dapat dipakai karena durasi hubungan *Finish-to-Start* sudah dikonversi menjadi durasi *Lag* dari hubungan *Start-to-Start*. Maka dari itu, Persamaan (2) akan dimodifikasi menjadi persamaan untuk hubungan *Start-to-Start*. Persamaan (2) berubah menjadi Persamaan (8) berikut,

$$ST_i + LT_{SSi} \leq ST_j \quad (8)$$

ST_i adalah *Start Time* dari aktivitas *i* (sebelum), LT_{SSi} adalah *Lag Time* dari hubungan *Start-to-Start* aktivitas *i*, dan ST_j adalah *Start Time* dari aktivitas *j* (setelah). Persamaan (8) menunjukkan *Start Time* aktivitas *j* hanya bisa dimulai setelah aktivitas *i* ditambahkan dengan durasi *lag*-nya selesai atau lebih.

Start Time dalam variabel solusi dibuat secara acak. Namun untuk membatasi agar hasil acak tidak melanggar banyak *constraints*, maka dibuat formula baru untuk membatasi *Start Time* yang dibuat. Formula dapat dilihat pada Persamaan (9) berikut,

$$ST_j = maxST_i + \{ rand(0-1) \times (maxFT_i - maxST_i) \} \quad (9)$$

ST_j merupakan aktivitas setelah aktivitas *i*, $maxST_i$ merupakan syarat mulai aktivitas *j* dari aktivitas *i* terbelakang, $rand(0-1)$ merupakan angka acak dari nol sampai satu, $maxFT_i$ merupakan *Finish Time* dari aktivitas *i* terbelakang. $maxFT_i - maxST_i$ adalah sisa waktu dari $maxST_i$ sampai $maxFT_i$. Sistem acak terdapat pada bagian $rand(0-1) \times (maxFT_i - maxST_i)$. Dengan adanya rumus ini program dapat mengacak *Start Time* tanpa melanggar *Constraints* dan memberikan keleluasan terhadap algoritma metaheuristik dalam memodifikasi data.

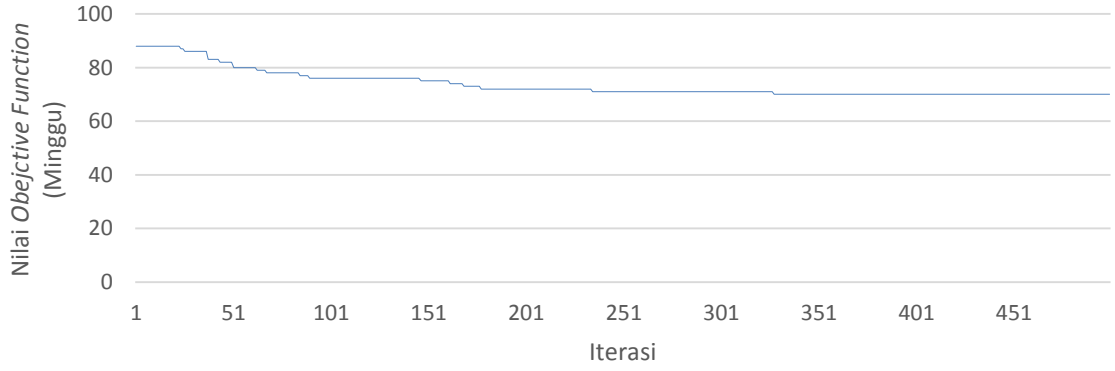
Solusi yang diolah dalam formulasi modifikasi algoritma SOS dibatasi dengan *lower bound* dan *upper bound* agar pencarian dapat lebih cepat dan akurat. *Lower bound* yang ditentukan adalah menggunakan *early start. Upper bound* yang ditentukan adalah total semua durasi aktivitas, yaitu 193 minggu. Batasan ini ditentukan agar program tidak menggeser *start time* lebih jauh yang dapat mengakibatkan *total duration* semakin panjang dan memiliki banyak *slack* atau waktu kosong.

4.4 Hasil Analisis Data

Percobaan dilakukan sebanyak 3 kali sesuai dengan keterbatasan yang sudah ditentukan di atas. Percobaan program menghasilkan nilai *objective function*, yaitu *total duration*, dan variabel solusi ynag berisi set *start time* terbaik. Berikut hasil dari ketiga percobaan.

a. Percobaan Pembatasan *Resource* 75% dari Mula-mula

Percobaan pertama dilakukan dengan membatasi *resource* untuk pekerjaan struktur menjadi 468 orang/minggu dan *resource* untuk pekerjaan arsitektur menjadi 536 orang/minggu. Grafik konvergensi dari hasil pencarian program dapat dilihat pada Gambar 6.

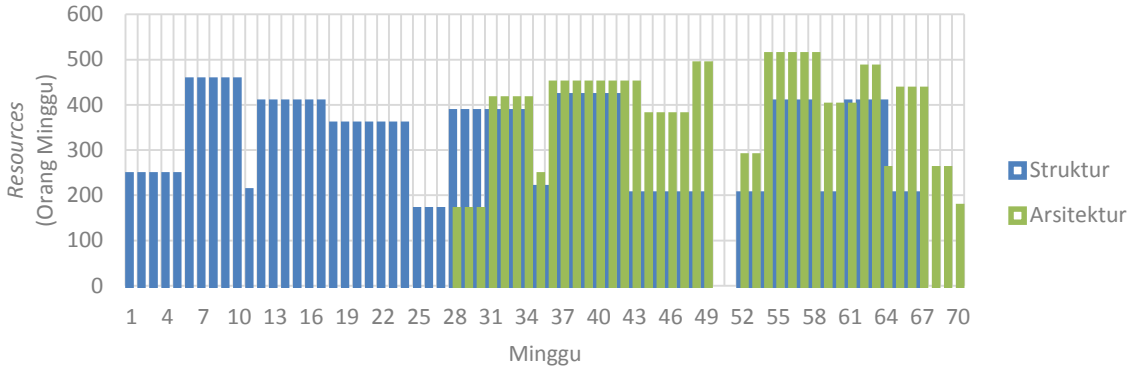


Gambar 6. Grafik Konvergensi *Fitness* Rk=75%

Hasil optimal dan konvergen didapatkan pada percobaan dengan iterasi 500. *Total Duration* optimal yang didapatkan adalah 70 minggu yang awalnya 59 minggu dan sudah konvergen pada iterasi ke 328 dan seterusnya. Hasil pengoptimasian dapat dilihat pada Tabel 2 dan pembatasan *resource* dapat dilihat pada Gambar 7.

Tabel 2. Pengoptimasian *Start Time* Rk=75%

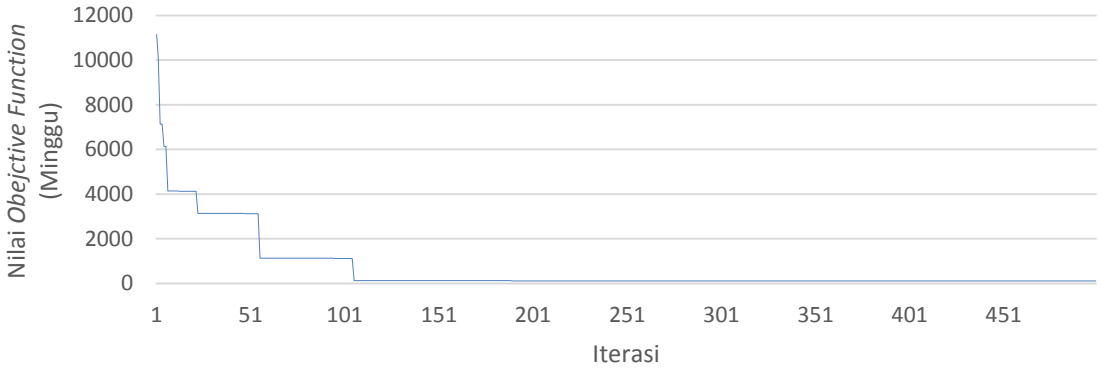
Aktivitas	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ST Sebelum	1	6	14	11	21	24	31	36	41	46	24	27	31	34	36	40	42	46	49	54
ST Sesudah	1	6	18	12	25	28	37	47	55	61	28	31	36	62	44	48	52	56	62	65



Gambar 7. *Resource Diagram* Rk=75%

b. Percobaan Pembatasan *Resource* 50% dari Mula-mula

Percobaan kedua dilakukan dengan membatasi *resource* untuk pekerjaan struktur menjadi 312 orang/minggu dan *resource* untuk pekerjaan arsitektur menjadi 357 orang/minggu. Grafik konvergensi dari hasil pencarian program dapat dilihat pada Gambar 8.

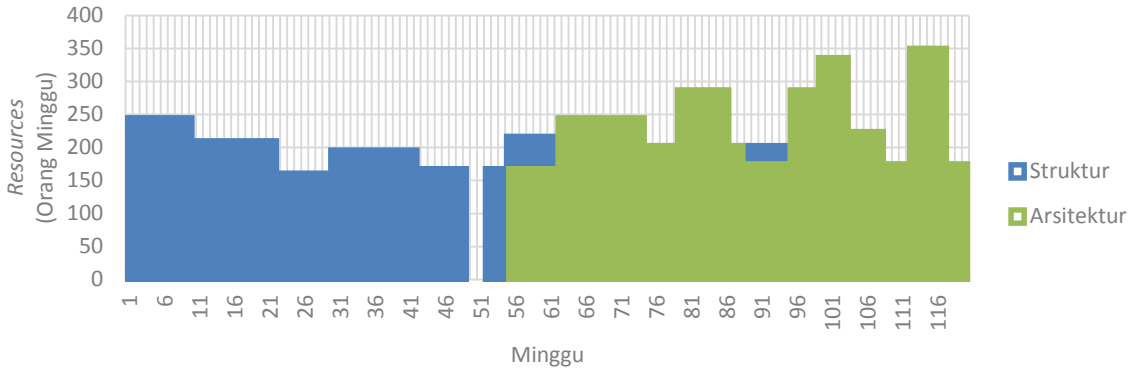


Gambar 8. Grafik Konvergensi *Fitness* Rk=80%

Hasil optimal dan konvergen didapatkan pada percobaan dengan iterasi 500. *Total Duration* optimal yang didapatkan adalah 120 minggu yang awalnya 59 minggu dan sudah konvergen pada iterasi ke 235 dan seterusnya. Hasil pengoptimasian *Start Time* dapat dilihat pada Tabel 3 dan pembatasan *resource* dapat dilihat pada Gambar 9.

Tabel 3. Pengoptimasian *Start Time* Rk=50%

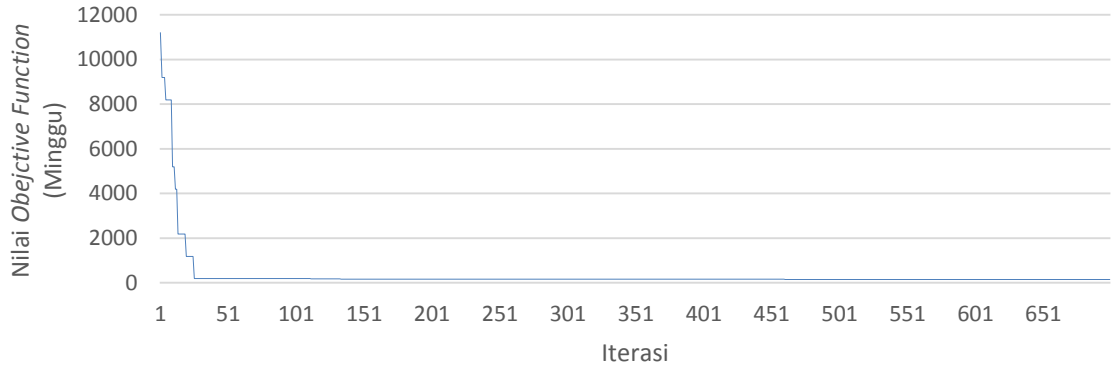
Aktivitas	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ST Sebelum	1	6	14	11	21	24	31	36	41	46	24	27	31	34	36	40	42	46	49	54
ST Sesudah	1	11	23	30	43	55	70	80	90	100	55	62	75	79	89	95	99	109	112	115



Gambar 9. *Resource Diagram* Rk=50%

c. Percobaan Pembatasan *Resource* paling Minimal dari Mula-mula

Percobaan ketiga dilakukan dengan membatasi *resource* untuk pekerjaan struktur menjadi 245 orang/minggu dan *resource* untuk pekerjaan arsitektur menjadi 245 orang/minggu. Grafik konvergensi dari hasil pencarian program dapat dilihat pada Gambar 10.

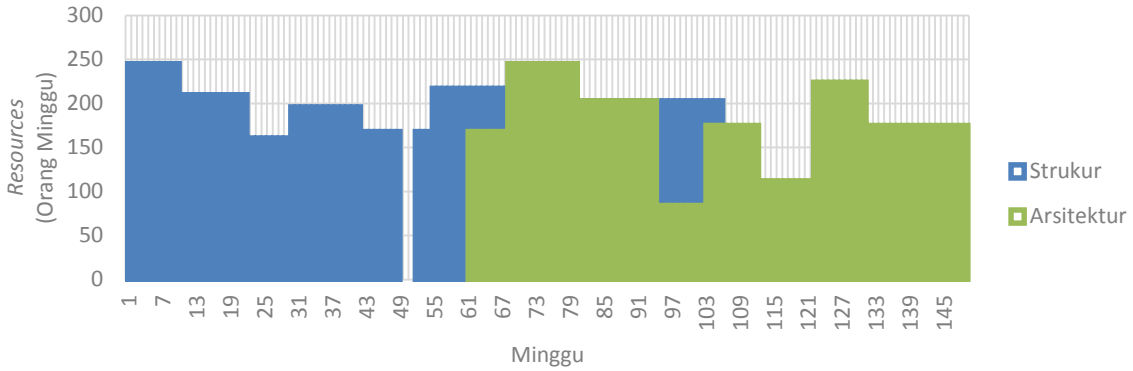


Gambar 10 Grafik Konvervengsi *Fitness* Rk Minimal

Hasil optimal dan konvergen didapatkan pada percobaan dengan iterasi 700. *Total Duration* optimal yang didapatkan adalah 149 minggu yang awalnya 59 minggu dan sudah konvergen pada iterasi ke 588 dan seterusnya. Hasil pengoptimasian *Start Time* dapat dilihat pada Tabel 4 dan pembatasan *resource* dapat dilihat pada Gambar 11.

Tabel 4. Pengoptimasian *Start Time* Rk Minimal

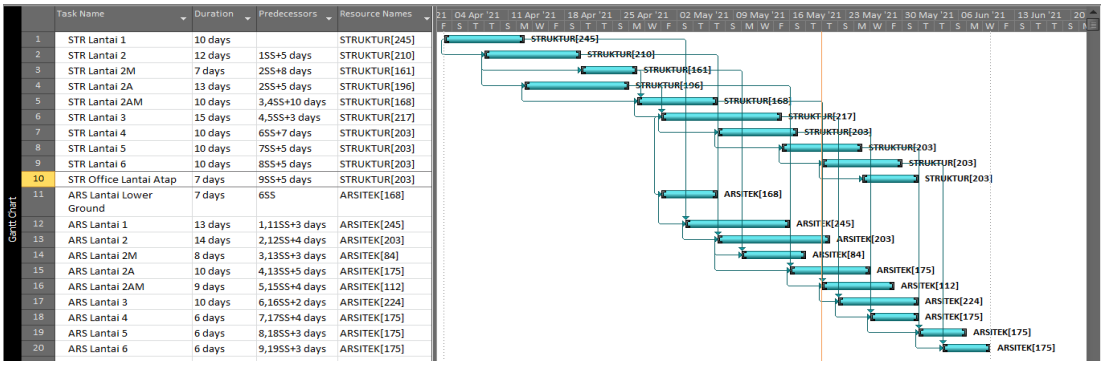
Aktivitas	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ST Sebelum	1	6	14	11	21	24	31	36	41	46	24	27	31	34	36	40	42	46	49	54
ST Sesudah	1	11	23	30	43	55	70	80	90	100	61	68	81	95	103	113	122	132	138	144



Gambar 11. *Resource Diagram* Rk Minimal

4.5 Perbandingan Hasil SOS dengan *Microsoft Project* 2010

Program *Microsoft Project* 2010 mempunyai fitur *Resource Levelling* untuk meratakan kebutuhan *resource*. Namun dalam penelitian ini, fitur tersebut juga dapat digunakan untuk menyelesaikan permasalahan keterbatasan *resource*. Data-data proyek kemudian dimasukkan ke dalam program, dan tampilan data-data yang dimasukkan dapat dilihat pada Gambar 12.



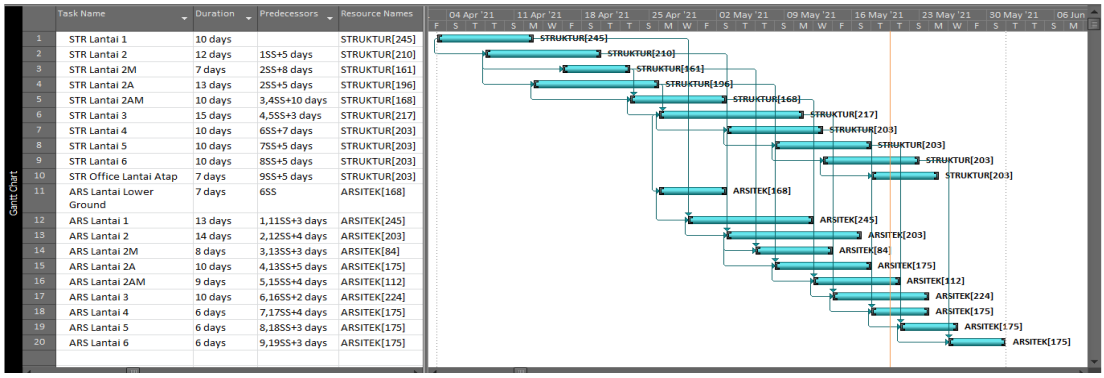
	Task Name	Duration	Predecessors	Resource Names
1	STR Lantai 1	10 days		STRUKTUR[245]
2	STR Lantai 2	12 days	1SS+5 days	STRUKTUR[210]
3	STR Lantai 2M	7 days	2SS+8 days	STRUKTUR[161]
4	STR Lantai 2A	13 days	2SS+5 days	STRUKTUR[196]
5	STR Lantai 2AM	10 days	3,4SS+10 days	STRUKTUR[168]
6	STR Lantai 3	15 days	4,5SS+3 days	STRUKTUR[217]
7	STR Lantai 4	10 days	6SS+7 days	STRUKTUR[203]
8	STR Lantai 5	10 days	7SS+5 days	STRUKTUR[203]
9	STR Lantai 6	10 days	8SS+5 days	STRUKTUR[203]
10	STR Office Lantai Atap	7 days	9SS+5 days	STRUKTUR[203]
11	ARS Lantai Lower Ground	7 days	6SS	ARSITEK[168]
12	ARS Lantai 1	13 days	1,11SS+3 days	ARSITEK[245]
13	ARS Lantai 2	14 days	2,12SS+4 days	ARSITEK[203]
14	ARS Lantai 2M	8 days	3,13SS+3 days	ARSITEK[84]
15	ARS Lantai 2A	10 days	4,13SS+5 days	ARSITEK[175]
16	ARS Lantai 2AM	9 days	5,15SS+4 days	ARSITEK[112]
17	ARS Lantai 3	10 days	6,16SS+2 days	ARSITEK[224]
18	ARS Lantai 4	6 days	7,17SS+4 days	ARSITEK[175]
19	ARS Lantai 5	6 days	8,18SS+3 days	ARSITEK[175]
20	ARS Lantai 6	6 days	9,19SS+3 days	ARSITEK[175]

Gambar 12. *Input* Data Program *Microsoft Project* 2010

a. Perbandingan Pengoptimasian dengan Rk=75% dari Mula-mula

Pembatasan *resource* dengan pekerjaan struktur 468 orang/hari dan pekerjaan arsitektur 536 orang/hari dimasukkan ke program *Microsoft Project* 2010 dan mendapatkan hasil seperti pada Gambar 13.

Hasil pada Gambar 13 dapat terlihat aktivitas diundur untuk memenuhi kebutuhan *resource* yang ditentukan. Perbandingan hasil dari pengoptimasian menggunakan SOS dan *Microsoft Project* 2010 dapat dilihat pada Tabel 5.



Gambar 13. Hasil *Microsoft Project Resource Levelling Solver* Rk=75%

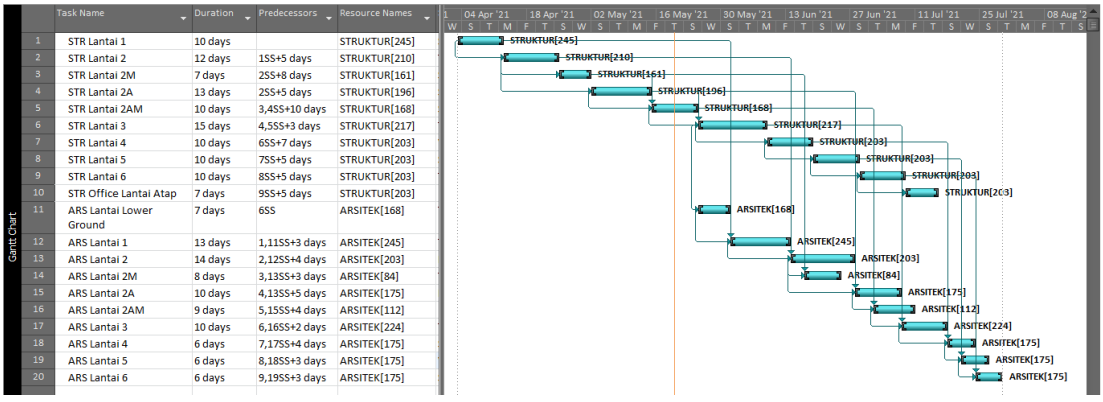
Tabel 5. Perbandingan Pengoptimasian dengan Rk=75%

Aktivitas	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
ST Sebelum	1	6	14	11	21	24	31	36	41	46	24	27	31	34	36	40	42	46	49	54
SOS	1	6	18	12	25	28	37	47	55	61	28	31	36	62	44	48	52	56	62	65
MS Project	1	6	18	11	25	28	35	43	48	55	28	31	35	38	44	48	50	56	62	65

Dari hasil perhitungan SOS dan *solver* dari *Microsoft Project* 2010 didapatkan *Total Duration* yang sama, yaitu 70 minggu. Keduanya memiliki beberapa hasil pengoptimasian *Start Time* yang berbeda pada aktivitas 7, 8, 9, 10, 14 dan 17 berdasarkan Tabel 4. Hasil SOS juga terlihat cukup acak karna memiliki satu aktivitas yang berada di belakang pada aktivitas 14. Berdasarkan dari kedua hasil dapat disimpulkan SOS dan *Microsoft Project* 2010 dapat menyelesaikan permasalahan keterbatasan *resource* secara optimal dengan hasil yang sama.

b. Perbandingan Pengoptimasian dengan Rk=50% dari Mula-mula

Pembatasan *resource* dengan pekerjaan struktur 312 orang/hari dan pekerjaan arsitektur 357 orang/hari dimasukkan ke program *Microsoft Project* 2010 dan mendapatkan hasil seperti pada Gambar 14.



Gambar 14. Hasil *Microsoft Project Resource Levelling Solver* Rk=50%

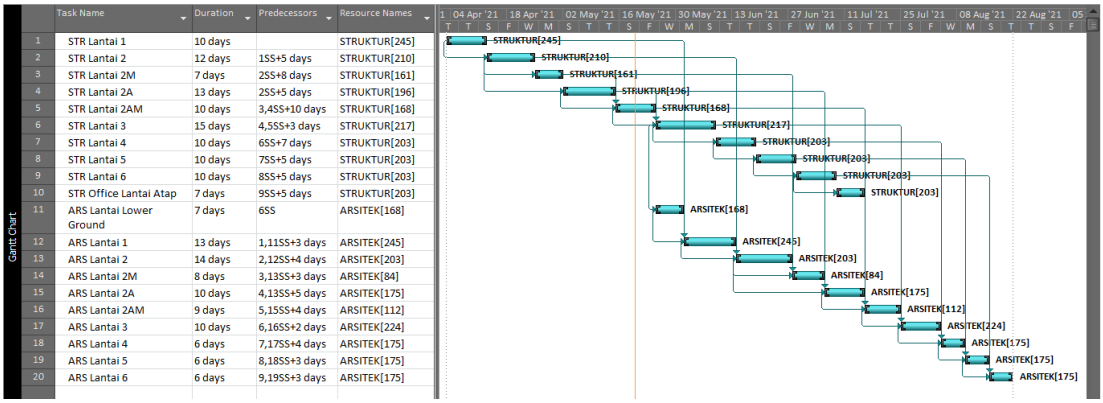
Hasil pada Gambar 14 dapat terlihat semua aktivitas harus diundur untuk memenuhi kebutuhan *resource* yang ditentukan. Aktivitas-aktivitas pada pekerjaan struktur yang awalnya memiliki hubungan *Start-to-Start* harus dimundurkan menjadi hubungan *Finish-to-Start* agar

dapat memenuhi pembatasan *resource* yang ditentukan. Perbandingan hasil dari pengoptimasian menggunakan SOS dan *Microsoft Project* 2010 dapat dilihat pada Tabel 6.

Tabel 6. Perbandingan Pengoptimasian dengan Rk=50%

Aktivitas	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ST Sebelum	1	6	14	11	21	24	31	36	41	46	24	27	31	34	36	40	42	46	49	54
SOS	1	11	23	30	43	55	70	80	90	100	55	62	75	79	89	95	99	109	112	115
MS Project	1	11	23	30	43	55	70	80	90	100	55	62	75	78	89	94	99	109	112	115

Pada Tabel 6 terlihat hasil pengoptimasian dari SOS dan dari *Resource Levelling Microsoft Project* 2010 memiliki *Total Duration* yang sama, yaitu 120 minggu. Pada Tabel 6 terlihat ada satu perbedaan *Start Time* pada aktivitas 16. Pada aktivitas 16 memiliki *lag* sehingga *Start Time* aktivitas tersebut dapat bervariasi.



Gambar 15 Hasil *Microsoft Project Resource Levelling Solver* Rk Minimal

c. Perbandingan Pengoptimasian dengan Rk=75% dari Mula-mula

Pembatasan *resource* dengan Pekerjaan Struktur 245 orang/hari dan Pekerjaan Arsitektur 245 orang/hari dimasukkan ke program *Microsoft Project* 2010 dan mendapatkan hasil seperti pada Gambar 15.

Pada Gambar 15 terlihat hasil *solver* dari *Microsoft Project* 2010 memaksakan semua aktivitas menjadi *Finish-to-Start* untuk memenuhi keterbatasan *resource* yang sudah ditentukan. Program tidak dapat mencari aktivitas yang dapat berjalan secara bersamaan berdasarkan keterbatasan *resource* yang ditentukan. Itu berarti hanya satu aktivitas yang dapat berjalan sendiri untuk berdasarkan keterbatasan *resource*. Hasil pengoptimasian *Start Time* kemudian dimasukkan ke dalam Tabel 7 untuk melihat perbandingan dengan hasil dari pengoptimasian menggunakan SOS dan *Microsoft Project* 2010.

Tabel 7 Perbandingan Pengoptimasian dengan Rk minimal

Aktivitas	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ST Sebelum	1	6	14	11	21	24	31	36	41	46	24	27	31	34	36	40	42	46	49	54
SOS	1	11	23	30	43	55	70	80	90	100	61	68	81	95	103	113	122	132	138	144
MS Project	1	11	23	30	43	55	70	80	90	100	55	62	75	89	97	107	116	126	132	138

Pada Tabel 7 terlihat hasil pengoptimasian dari SOS dan *Microsoft Project* 2010 memiliki *Start Time* yang sama pada Aktivitas Pekerjaan Struktur, namun memiliki perbedaan pada Aktivitas Pekerjaan Arsitektur. Perbedaannya terletak pada Aktivitas 11 yang mempunyai *Predecessor* dengan Aktivitas 6 dengan hubungan *Start-to-Start* tanpa adanya *Slack*. Dari hasil pencarian SOS, *Start Time* Aktivitas 11 yang didapatkan lebih banyak 6 minggu dari hasil program *Microsoft Project* 2010. Faktor pencarian acak dari SOS membuat SOS mendapatkan jawaban seperti di atas. Setelah itu, semua aktivitas, baik dari pencarian SOS maupun *Microsoft Project* 2010, memaksa semua aktivitas untuk menjadi *Finish-to-Start* agar proyek dapat berjalan dengan keterbatasan *resource* yang ditentukan. *Total Duration* yang didapatkan dari SOS adalah 149 minggu dan dari program *Microsoft Project* 2010 adalah 143 minggu.

Kecepatan pencarian dari algoritma SOS dipengaruhi oleh banyaknya iterasi, variabel solusi, dan faktor spesifikasi komputer yang dipakai. Semakin banyak jumlah iterasi dan variabel solusi, maka pencarian dapat semakin lama. Pencarian dengan penambahan jumlah iterasi memiliki hubungan linear dengan waktu pencarian. Setiap 100 iterasi, waktu pencarian bertambah ±9,67 detik.

Untuk penggunaan fungsi *Resource Levelling* pada program *Microsoft Project* 2010, hasil yang didapatkan tidak langsung optimal. Program memberikan jawaban yang memiliki aktivitas-aktivitas yang melanggar *precedence constraints* aktivitas. Jadi perlu mengatur ulang program dengan *respect links* agar program kembali mengutamakan *predecessor* masing-masing aktivitas. Setelah pengaturan *respect links* ini dilakukan, program kembali mengatur aktivitas berdasarkan *predecessor*, namun hasilnya tidak memenuhi keterbatasan *resource*. Lalu fungsi *Resource Levelling* kembali digunakan, dan mengatur kembali setiap aktivitas untuk *respect links*. Proses ini terus diulang sampai mendapatkan hasil yang optimal. Pada saat mengaktifkan fungsi *Resource Levelling*, program tidak memiliki fungsi untuk tetap mempertahankan *precedence* dari setiap aktivitas, sehingga program dapat melanggar *constraint* yang telah ditentukan.

Berdasarkan percobaan menggunakan algoritma SOS dan *solver resouce levelling* dari program *Microsoft Project* 2010 dengan pembatasan *resource* yang sama, algoritma SOS memiliki hasil yang sama dengan *solver* dari *Microsoft Project* 2010. Kecenderungan *solver* dari program *Microsoft Project* 2010 memilih untuk mengatur penjadwalan berdasarkan urutan identitas pekerjaan, sedangkan algoritma SOS dengan keunikan dari algoritma metaheuristik mendapatkan jawaban secara acak. Proses pencarian algortima SOS cukup lambat karena menunggu proses iterasi untuk dapat menemukan jawaban yang optimal, namun langsung mendapatkan jawaban yang optimal. S*olver* dari program *Microsoft Project* 2010 membutuhkan proses yang cukup rumit dan lebih lama karena harus mengulang proses *solver* sampai menemukan jawaban yang optimal.

5. KESIMPULAN DAN SARAN

Dalam percobaan-percobaan dengan pembatasan *resource* yang ditentukan di awal dan mendapatkan hasil yang kemudian dibandingkan antara hasil dari algoritma SOS dan *Microsoft Project* 2010, maka dapat disimpulkan hal-hal berikut:

a. Algoritma SOS terbukti mampu menyelesaikan permasalahan RCPSP dengan menggunakan *objective function* dan modifikasi *constraints* yang disesuaikan dengan permasalahan dalam proyek yang diteliti.
b. Hasil pengoptimasian algoritma SOS memiliki *total duration* yang sama dengan *solver* dari program *Microsoft Project* 2010, kecuali dalam percobaan terakhir dengan pembatasan *resource* paling minimal. Proses pencarian algortima SOS cukup lambat karena menunggu proses iterasi untuk dapat menemukan jawaban yang optimal, namun langsung mendapatkan jawaban yang optimal. S*olver* dari program *Microsoft Project* 2010 membutuhkan proses yang cukup rumit karena harus diulang sampai menemukan jawaban yang optimal.

Algoritma SOS dapat dipakai untuk menyelesaikan permasalahan RCPSP yang lebih kompleks dengan tambahan modifikasi program. Untuk penelitian berikutnya dapat menambahkan variabel solusi pencari agar dapat membantu pencarian yang lebih akurat.

6. DAFTAR REFERENSI

Cheng, M. Y., Prayogo, D., & Tran, Duc-Hoc (2016). “Optimizing Multiple Resources Leveling in Multiple Projects Using Discrete Symbiotic Organisms Search”. *Journal of Computing in Civil Engineering*, 30(3), 1–9.

Cheng, M. Y., & Prayogo, D. (2014). “Symbiotic Organisms Search: a New Metaheuristic Optimization Algorithm”. *Computers and Structures*, 139, 98–112.

Desale, S., Rasool, A., Andhale, S., & Rane, P. (2015). "Heuristic and Meta-Heuristic Algorithms and Their Relevance to the Real World: A Survey". *International Journal of Computer Engineering in Research Trends*, *351*(5), 2349–7084. Retrieved from http://www.ijcert.org

El-abbasy, M. S., Elazouni, A., & Zayed, T. (2016). “Automation in Construction MOSCOPEA : Multi-Objective Construction Scheduling Optimization Using Elitist Non-Dominated Sorting Genetic Algorithm”. *Automation in Construction*, 71, 153–170.

Koulinas, G., Kotsikas, L., & Anagnostopoulos, K. (2014). “A Particle Swarm Optimization Based Hyper-Heuristic Algorithm for the Classic Resource Constrained Project Scheduling Problem”. *Information Sciences*, 277, 680–693.

Vanhoucke, M. (2018). “A Tool to Test and Validate Algorithms for the Resource-Constrained Project Scheduling Problem”. *Computers & Industrial Engineering*.

Zhang, H., Li, X., Li, H., & Huang, F. (2005). *Particle Swarm Optimization-Based Schemes for Resource-Constrained Project Scheduling*, 14, 393–404.

Zhang, H., & Tam, C. M. (2006). *Project Particle Swarm Optimization for Resource-Constrained Project Scheduling*, 24, 83–92.