UNIVERSITAS INDONESIA

MODEL PENUGASAN *DUE DATE* MULTI ITEM MULTI-LEVEL PADA *JOB SHOP* DINAMIS MESIN PARALEL DENGAN MEMPERHATIKAN *DEFECT RATE*

TESIS

PURDIANTA
1006735290

FAKULTAS TEKNIK
PROGRAM TEKNIK INDUSTRI
JAKARTA
DESEMBER 2011

UNIVERSITAS INDONESIA

MODEL PENUGASAN *DUE DATE* MULTI ITEM MULTI-LEVEL PADA *JOB SHOP* DINAMIS MESIN PARALEL DENGAN MEMPERHATIKAN *DEFECT RATE*

TESIS

Diajukan sebagai salah satu syarat untuk memperoleh gelar Magister Teknik

PURDIANTA
1006735290

FAKULTAS TEKNIK
PROGRAM TEKNIK INDUSTRI
JAKARTA
DESEMBER 2011

HALAMAN PERNYATAAN ORISINALITAS

Skripsi/Tesis/Disertasi ini adalah hasil karya saya sendiri,
dan semua sumber baik yang dikutip maupun dirujuk
telah saya nyatakan dengan benar.

Nama : Purdianta

NPM : 1006735290

Tanda Tangan : .....

Tanggal : 21 Desember 2011

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh :

Nama : Purdianta

NPM : 1006735290

Program Studi : Teknik Industri

Judul Skripsi : Model Penugasan *Due Date* Multi Item Multi-Level Pada *Job Shop* Dinamis Mesin Paralel Dengan Memperhatikan *Defect Rate*

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Magister Teknik pada Program Studi Teknik Industri, Fakultas Teknik, Universitas Indonesia.

DEWAN PENGUJI

Pembimbing 1 : Ir. Isti Surjandari P.MT., MA., Ph.D. (.....)

Pembimbing 2 : Ir. Amar Rachman, MEIM (.....)

Penguji 1 : Prof. Dr. Ir.T. Yuri. M.Z.,MEngSc (.....)

Penguji 2 : Ir. Sri Bintang P., MSISE, Ph.D (.....)

Penguji 3 : Dr. Ir. Akhmad Hadiyatno, MBT (.....)

Penguji 4 : Ir. Djoko S. Gabriel, MT (.....)

Ditetapkan di : Salemba

KATA PENGANTAR

Dengan mengucapkan puji dan syukur kehadirat Tuhan Yang Maha Esa yang senantiasa memberikan rahmat-Nya, sehingga penulis dapat menyelesaikan tesis ini. Adapun tesis ini diajukan untuk memenuhi salah satu persyaratan untuk memperoleh gelar Magister Teknik jenjang pendidikan strata-2 Program Studi Teknik Industri pada Universitas Indonesia.

Keberhasilan penyusunan tesis ini tidak terlepas dari dukungan, bantuan dan bimbingan dari berbagai pihak. Untuk itu penulis menyampaikan penghargaan dan rasa terima kasih yang sebesar-besarnya kepada yang terhormat:

1. Ir. Isti Surjandari P.MT., MA., Ph.D, selaku pembimbing yang telah banyak memberikan pengarahan, waktu dan dukungan serta keakraban selama masa kuliah.
2. Ir. Amar Rachman, MEIM, selaku dosen pembimbing yang telah banyak memberi bantuan, masukan dan bimbingan yang berharga bagi penulis.
3. Segenap Dosen Departemen Teknik Industri, yang telah banyak memberikan bimbingan dan ilmu yang sangat berharga bagi penulis.
4. Seluruh staff Departemen Teknik Industri, yang telah banyak membantu dalam masalah administrasi.
5. Papa dan kakak-kakak yang selalu memberikan dukungan moril maupun materil, motivasi dan doa kepada penulis.
6. Rekan-rekan Magister Teknik Industri Angkatan 2010, terima kasih atas keakraban dan kerja samanya.
7. Semua pihak yang tidak dapat penulis sebutkan satu persatu yang telah banyak membantu dalam penyusunan laporan tugas akhir ini.

Penulis menyadari bahwa tesis ini masih jauh dari sempurna. Karena itu, kritik dan saran yang membangun sangat penulis harapkan. Penulis berharap tesis ini dapat memberikan manfaat bagi semua pihak yang membacanya.

Jakarta, 21 Desember 2011

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama : Purdianta
NPM : 1006735290
Program Studi : Teknik Industri
Departemen : Teknik Industri
Fakultas : Teknik
Jenis karya : Tesis

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty-Free Right*)** atas karya ilmiah saya yang berjudul :

MODEL PENUGASAN *DUE DATE* MULTI ITEM MULTI-LEVEL PADA *JOB SHOP* DINAMIS MESIN PARALEL DENGAN MEMPERHATIKAN *DEFECT RATE*

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Nonekslusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/ formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan tugas akhir saya tanpa meminta izin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Salemba
Pada tanggal : 21 Desember 2011
Yang menyatakan

(Purdianta)

ABSTRAK

Nama : Purdianta
Program Studi : Teknik Industri
Judul : Model Penugasan *Due Date* Multi Item Multi-Level Pada *Job Shop* Dinamis Mesin Paralel *Dengan* Memperhatikan *Defect Rate*

Penelitian ini, bertujuan mengembangkan model penentuan *due date* melalui penjadwalan *batch* untuk melakukan pemenuhan (Model 1) dan penentuan (Model 2) *due date* dengan mempertimbangkan *defect rate*. Pada sistem produksi *job shop* dinamis mesin parallel yang memproduksi muti-item berstruktur multi-level. Ukuran performansi yang digunakan yaitu *total actual flow time*. Proses penjadwalan dilakukan dengan menggunakan teknik penyisipan (*insertion technique*), yaitu melakukan penyisipan operasi-operasi disemua posisi pemproses yang mungkin pada semua mesin yang tersedia. Pemilihan posisi didasarkan pada kriteria tertentu dengan memperhatikan terpenuhinya semua urutan proses (*routing*) dan hubungan proses pendahulu yang ada diantara setiap operasi. Permasalahan yang diselesaikan dalam penelitian mencakup kondisi statis dan dinamis.

Keyword : *due date, total actual flow time, defect rate, job shop,* teknik penyisipan

ABSTRACT

Name : Purdianta
Study Program : Master of Industrial Engineering
Title : Due Date Assignment Multi-Item Multi-Level Model in Dynamic Job Shop Parallel Machines by Considering Defect Rate

This research, aims to develop due date determination model trough batch scheduling to accomplish the due date (Model 1) and due date assignment (Model 2) with defect rate consideration. On dynamic job shop machines parallel that produce multi- item structured multi-level. The measurement of performance used is the total actual flow time. Scheduling process is done by using the insertion technique, perform insertion operation at all position that may be available on all machine. The selection criteria are based on a specific criteria with respect to fulfillment of all the process sequence and predecssoe existing between each operation. The problem are solved in the static and dynamic conditions.

Keyword :due date, total actual flow time, defect rate, job shop, insertion technique

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Seiring dengan pertumbuhan dunia industri, menyebabkan semakin ketatnya persaingan yang harus dihadapi. Salah satu faktor yang perlu diperhatikan adalah kemampuan untuk secara cepat merespon kebutuhan konsumen atau pelanggan. Kondisi mengakibatkan terjadinya peralihan dari *mass production* menjadi *mass customization*. Tentunya, agar dapat merespon permintaan pelanggan secara cepat perlu disiapkan sejumlah persediaan yang cukup besar, supaya tersedia pada saat dibutuhkan.

Namun, disisi lain untuk meminimasi lamanya suatu pekerjaan berada di lantai pabrik (*shop time*) justru perlu dilakukan dengan meminimasi jumlah persediaan seperti yang dikemukakan oleh Baker (1974), Karmarkar (1987) dan Halim (1993). Dalam penelitian ini juga menyatakan bahwa ketepatan pemenuhan *due-date* cenderung dianggap lebih penting dari pada meminimasi lamanya pesanan berada di lantai pabrik. Hal ini terjadi karena pemenuhan *due-date* secara tepat lebih terkait dengan kepuasaan pelanggan (*customer satisfaction*).

Sistem produksi tepat waktu (*Just In Time*) merupakan sistem produksi yang dapat mengakomodir kondisi ini. Di mana pada satu sisi bertujuan melakukan pemenuhan ketepatan *due-date* dan disisi lain juga berusaha untuk meminimasi jumlah persediaan. Karena konsep sistem produksi tepat waktu bertujuan untuk menghasilkan barang atau produk pada waktu dan kuantitas yang tepat serta kualitasnya, Vollmann (2005).

Penelitian tentang penjadwalan *batch* telah dilakukan oleh Dobson et al.(1987,1989) dengan menggunakan pendekatan maju tetapi belum mempertimbangkan *due-date* sebagai faktor pembatas. Model penjadwalan yang menggunakan konsep tepat waktu dikembangkan oleh Halim dan Ohta (1993) yaitu model penjadwalan *batch* yang memasukan pembatas bahwa tidak diperbolehkan adanya pekerjaan yang terlambat. Halim dan Barnali (1998) mengembangkan model penjadwalan *batch* pada sistem produksi *flow shop* untuk kondisi permintaan dinamis dan memproduksi multi-item dengan multi *due-date* dengan tujuan meminimasi total waktu tinggal aktual. Dalam penelitian tidak

dibahas adanya ketergantungan antara komponen satu dengan komponen lainnya dalam membentuk produk akhir sesuai dengan struktur produknya.

Pengembangan algoritma heuristik untuk masalah penjadwalan *batch* melalui teknik penyisipan pada lingkungan *job shop* dengan waktu *set-up* independen, (Sotskov et al.,1999). Di mana setiap pekerjaan memiliki urutan proses pada masing-masing mesin yang tidak dapat dipertukarkan. Dengan tujuan untuk menimimasi *completion time*, melalui teknik penyisipan (*insertion technique*) memungkinkan untuk melakukan penyisipan pekerjaan disepanjang waktu perencanaan dengan memperhatikan hubungan ketergantungan pekerjaan. Penelitian yang dilakukan terbatas pada kondisi *single machine*, dan belum memasukan unsur struktur produk (*Bill of Material-BOM*) didalam proses penjadwalan.

Pengembangan teknik penyisipan pada kondisi statis dan dinamis di dalam menyelesaikan permasalahan dengan keterbatasan sumber daya di dalam penjadwalan proyek telah dilakukan oleh Artigues et al.(2003). Dari hasil penelitian menunjukan bahwa teknik penyisipan memberikan penjadwalan yang lebih baik pada kondisi lingkungan dinamis. Selajutnya Vestjens et al (2007) melakukan kajian terhadap kompleksitas penyisipan pekerjaan dalam penjadwalan multi tahap dengan tujuan untuk menimasi *makespan* pada sistem produksi *flow shop*.

Mosheiov et al (2006) telah mengembangkan model penentuan *due-date* dan masalah penjadwalan aktivitas perawatan. Model yang dikembangkan bertujuan untuk menentukan urutan pekerjaan, *due-date* dan aktivitas perawatan. Adapun ukuran performansi yang digunakan meminimumkan total biaya pekerjaan yang diselesaikan lebih awal (*earliness*), total biaya keterlambatan (*tardiness*) dan total biaya penyelesaian pekerjaan tepat pada *due-date*nya.

Huang (2010), menggunakan pendekatan heuristik untuk menyelesaikan masalah *multi-objective job shop* dengan melakukan *lot-splitting* pada lingkungan JIT. Fungsi tujuan yang akan diminimasi total stok, mesin menunggu (*machine idle)*, dan biaya pengangkutan. Di mana biaya pengangkutan akan meningkat seiring dengan peningkatan jumlah *batch*, dari hasil penelitian ini dapat diketahui jumlah *batch* yang dapat memberikan total biaya minimum dari ketiga komponen

biaya yang sudah ditetapkan. Namun, dalam penelitian ini belum memasukan unsur *due date* baik dalam hal pemenuhan maupun penentuannya.

Penelitian penerapan genetik heuristik untuk meneliti masalah penjadwalan mesin pemprosesan paralel *batch* dengan ukuran pekerjaan yang berubah, di mana setiap mesin dapat memproses sekelompok pekerjaan secara bersamaan sebagai sebuah *batch* dengan tujuan meminimasi *makespan* (Kashan et al.,2008). Penelitian ini belum memasukan *due date* sebagai ukuran kinerja penjadwalan dan kedatangan pekerjaan yang mungkin terjadi sepanjang horizon waktu perencanaan.

Penentuan *due date* dan penjadwalan mesin paralel dengan penurunan pekerjaan, dengan tujuan untuk menimasi total *due date, earliness* dan *tardiness* penalti (Cheng et al.,2007). Dalam penelitian ini diasumsikan, di mana pekerjaan tergantung tingkat penurunan yang sama untuk semua pekerjaan.

Xia et al., (2008) mengembangkan suatu prosedur heuristik untuk menyelesaikan permasalahan pengurutan pekerjaan dan penentuan *due date* pada lingkungan JIT. Untuk meminimumkan kombinasi linier dari tiga jenis pinalti, yaitu penyelesaian lebih awal pekerjaan (*penalty on job earliness*), pinalti keterlambatan pekerjaan (*penalty on job tardiness*) dan pinalti penentuan *due date* yang panjang (*penalty associated with long due date assignment*). Dalam penelitian ini diasumsikan waktu proses pekerjaan tidak pasti, tetapi permasalahan yang dibahas pada kondisi mesin tunggal.

Penyelesaian permasalahan penjadwalan dan penentuan *due date* yang bertujuan untuk minimasi pekerjaan yang selesai lebih awal, terlambat, tertunda, penentuan *due date* dan biaya pengiriman atau transfer *batch* pada mesin tunggal, di mana *due date* dapat dikendalikan telah dilakukan oleh Shabtay, (2010). Dalam penelitian ini, strategi penentuan *due date* optimal sebagai fungsi dari pengurutan pekerjaan dan pembagian urutan pekerjaan kedalam *batch-batch*. Selain itu pendekatan penentuan *due date* yang digunakan berdasarkan *DIF (diferent due date) method*, sehingga setiap pekerjaan akan memiliki *due date* yang berbeda.

Penentuan *due date* dan penjadwalan pada lingkungan produk JIT dengan ukuran pekerjaan yang sama, di mana penelitian yang dilakukan untuk kondisi mesin tunggal dan mesin paralel dengan tujuan untuk meminimasi total bobot

keterlambatan dan penyelesaian lebih awal dari suatu pekerjaan dan biaya *due date* telah dilakukan oleh Tuong dan Soukhal, (2010). Pendekatan yang digunakan berdasarkan pada *CON* (*Common Due Date*) *method,* sehingga semua pekerjaan dianggap memiliki *due date* yang sama.

Pengembangan penentuan *due date* dan aturan keputusan penjadwalan pada lingkungan *job shop* dinamis, melalui pendekatan pemodelan dan analisa simulasi telah dilakukan oleh Vinod dan Sridharan (2011). Metode penentuan *due date* dikembangkan adalah pemprosesan dinamis ditambah waktu tunggu (*waiting time*), total pengerjaaan (*Total Work Content-TWK*), total waktu pengerjaan dinamis (*Dynamic Total Work Content-DTWK*) dan metode pengerjaan acak (*Random Work Content-RWK method*). Namun dalam penelitian ini, semua pekerjaan diasumsikan tersedia pada waktu sama dengan nol dan belum memperhatikan keterkaitan waktu penyelesaian antar komponen penyusun produk pada masing-masing tingkat.

Dari uraian tersebut diatas terlihat bahwa adanya kebutuhan untuk pengembangan model penjadwalan *batch* pada sistem produksi *job shop paralel* dinamis yang memproduksi multi item, di mana setiap item berstruktur multi level dengan memperhatikan *defect rate* yang dapat dihasilkan dari suatu proses produksi. Dengan tujuan meminimasi total waktu tinggal aktual pada lingkungan produksi *Just In Time* (JIT). Penggunakan teknik penyisipan dikarena metode ini lebih memberikan kemudahan untuk menyelesaikan permasalahan penjadwalan pada kondisi dinamis, melalui penyisipan pekerjaan disepanjangan horizon perencanaan yang mungkin dilakukan. Unsur *defect rate* perlu dipertimbangkan dalam proses penjadwalan karena memiliki pengaruh terhadap produk akhir yang akan dihasilkan, pada setiap tahapan proses produksi dan waktu penyelesaian pekerjaan.

1.2 Diagram Keterkaitan Masalah

Gambaran sistematis yang lebih menyeluruh mengenai keterkaitan permasalah dapat dilihat pada gambar 1.1, yang merupakan hubungan antar gejala permasalahan yang bentuk suatu permasalahan yang harus diselesaikan.

1.3 Perumusan Masalah

Penelitian ini mengembangkan model penjadwalan *batch* yang terdiri dari dua model. Model satu bertujuan melakukan penjadwalan untuk pemenuhan *due date* pada lingkungan *Job Shop Paralel*. Model dua bertujuan melakukan penjadwalan untuk penentuan *due date*. Pada sistem *Job Shop Pararel* dinamis yang memproses multi-item berstruktur multi-level, berdasarkan kinerja kualitas pada lingkungan produksi *Just In Time* (JIT). Pendekatan yang digunakan dalam penelitian ini adalah pendekatan mundur (*backward approach*) untuk pemenuhan *due date* dan pendekatan maju (*forward)* untuk penentuan *due date* serta teknik penyisipan dengan kriteria minimasi *total actual flow time*. Dengan memperhatikan tingkat cacat (*defect rate*) yang terjadi pada setiap tahapan proses yang dilalui.

1.4 Tujuan Penelitian

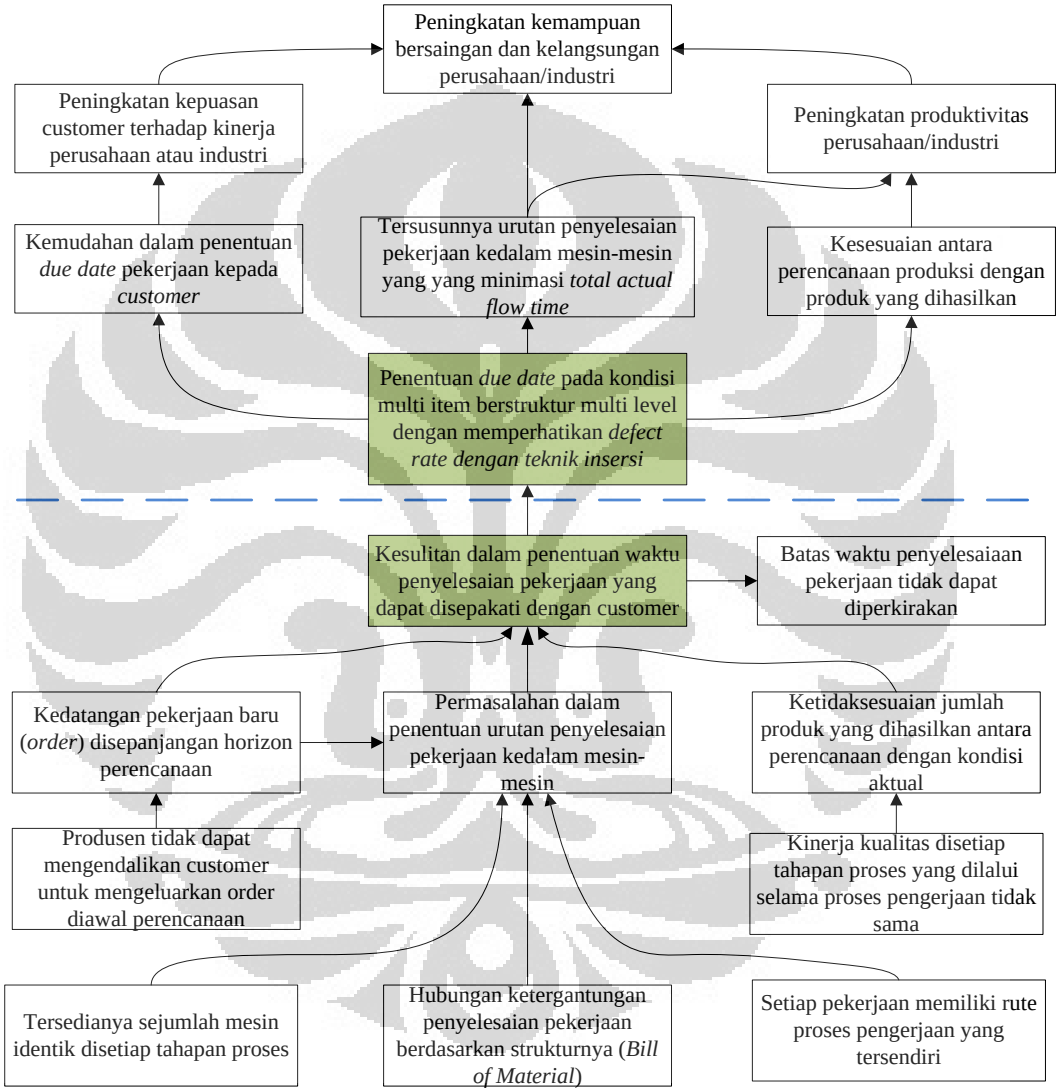
Tujuan dari penelitian ini adalah menghasilkan model penjadwalan *batch* untuk melakukan pemenuhan (Model satu) dan penentuan (Model dua) *due date* pada sistem produksi *job shop* dinamis mesin paralel yang memproses multi-item berstruktur multi level dan multi *due date*, dengan memperhatikan *defect rate*. Untuk mendapatkan *total actual flow time* yang minimum dengan menggunakan pendekatan teknik penyisipan. Di mana variable-variabel keputusan dalam model tersebut adalah jumlah *batch*, ukuran *batch*, dan urutan pemprosesan *batch* yang memberikan waktu tinggal aktual yang minimum.

1.5 Ruang Lingkup Pembahasan

Dalam penelitian ini, terdapat beberapa hal yang menunjukkan karakteristik masalah dalam penelitian tugas akhir ini adalah :

- *Defect rate* yang terjadi disetiap tahapan proses produksi bersifat deterministik (diketahui secara pasti).
- Waktu pemprosesan setiap operasi bersifat determistik.
- Sistem produksi yang ditinjau adalah sistem produksi *job shop*. Di mana masing-masing item memiliki urutan (*routing*) produksi dan struktur produksi sendiri yang berbeda dengan waktu *set-up* dan waktu proses diketahui secara pasti.

- Kedatangan pekerjaan (*order*) diasumsikan dinamis dalam arti setiap pekerjaan tidak harus datang pada saat waktu nol, tetapi dapat datang sepanjang horizon waktu perencanaan. Untuk setiap pekerjaan yang telah diterima tidak terjadi perubahan *due-date* pada masalah penjadwalan *batch*, perubahan perioritas, atau pun pembatalan pekerjaan .



Gambar 1. 1 Diagram Keterkaitan Masalah

- Setiap pekerjaan memiliki *due-date* tertentu yang harus dipenuhi dan tidak diperbolehkan terjadi keterlambatan pekerjaan. Pada permasalahan penjadwalan *batch* untuk pemenuhan *due date*.

- Setiap pekerjaan memiliki *due date* yang tidak diketahui dan dapat dinegosiasikan dengan konsumen, pada masalah penentuan *due date*.
- Waktu *set-up* yang terjadi pada setiap awal operasi tidak dimasukan kedalam waktu proses *batch*.

Untuk lebih spesifiknya, sistem produksi yang menjadi objek penelitian adalah jenis *job shop* yang memiliki karakteristik sebagai berikut;

- Terdapat beberapa tahapan proses produksi (*multi-stage*).
- Tiap *stage* hanya terdiri dari lebih dari satu unit mesin (*pararel*), dan
- Batas waktu penyelesaian pekerjaan (*due-date*) tidak sama (multi *due-date*) pada masalah pemenuhan *due date*.
- *Due date* dapat dinegosiasikan dengan konsumen, pada masalah penentuan *due date*.

1.6 Batasan Masalah

Untuk lebih memfokuskan pembahasan, ada beberapa asumsi yang digunakan dalam penelitian ini sebagai berikut.

- Ukuran *batch* dalam penelitian ini diasumsikan kontinu.
- Suatu mesin pada waktu tertentu hanya memproses satu buah *batch* operasi saja.
- *Set-up* mesin dapat dilakukan sebelum *batch* datang ke stasiun kerja.
- Selama proses operasi tidak terjadi gangguan pada fasilitas produksi.
- Setiap mesin memiliki fungsi khusus (*dedicated function*) yang tidak dipertukarkan dengan mesin lainnya.
- Waktu transportasi yang terjadi satu stasiun kerja ke stasiun kerja yang lainnya diabaikan, dan
- Waktu dan biaya *set-up* tidak dipengaruhi oleh besarnya ukuran *batch* dan urutan pengerjaan operasi.

1.7 Metodologi Penelitian

Penelitian ini merupakan pengembangan dari beberapa penelitian dibidang penjadwalan dan penentuan *due date* dilingkungan produksi JIT. Tujuan utama dari penelitian ini adalah menghasilkan suatu model penentuan *due date* melalui penjadwalan *batch* untuk *job shop* dinamis yang memproses multi-item

berstruktur multi-level dengan tujuan meminimasi *total actual flow time*. Di mana sistem produksi yang ditinjau berlingkungan just in time serta memperhatikan terjadinya tingkat cacat dari setiap tahap proses produksi, yang dinyatakan dengan persentase terjadinya cacat pada setiap proses yang dilalui. Adapun kerangka penelitian, sebagai berikut.

1. Studi Pendahuluan

   Penelitian ini dilakukan melalui studi literatur dan pengamatan terjadi dibeberapa industri berbasis *job shop* dengan kedatangan pekerjaan (*job)* bersifat dinamis.

2. Perumusan Masalah

   Dari berbagai permasalahan yang diperoleh dari studi literatur maupun permasalahan diketemukan di dunia industri, maka dirumuskan masalah yang akan diselesaikan dalam penelitian ini.

3. Tujuan Penelitian

   Dengan inti permasalahan yang ada di industri berbasis *job order* khususnya dan dilakukan studi literatur baik melaui jurnal internasional, laporan penelitian maupun buku teks, maka dirumuskan tujuan dilakukannya penelitian ini.

4. Pengembangan Model Kondisi Statis

   Proses pengembangan model penentuan *due date* untuk kondisi dinamis didalam penelitian ini, diawali dari pengembangan model untuk kondisi statis terlebih dahulu. Di mana pada model kondisi statis tidak kedatangan pekerjaan baru yang datang selama horizon perencanaan.

5. Pengembangan Model Kondisi Dinamis

   Model penentuan *due date* pada kondisi dinamis merupakan model yang menjadi tujuan penelitian ini. Di mana dalam kondisi dinamis izinkan terjadinya kedatangan pekerjaan baru di sepanjang horizon waktu perencanaan.

6. Pengujian dan Analisa Model

   Untuk memastikan model bekerja sesuai dengan tujuan yang diharapkan, maka dilakukan pengujian algoritma penyelesaian model baik pada kondisi statis maupun dinamis. Pengujian algoritma ini juga sebagai bagian dari proses verifikasi dan validasi model. Apabila model

yang dikembangkan belum dapat memberikan hasil yang diharapkan, maka akan dilakukan perbaikan terhadap algoritma penyelesaian.

7. Kesimpulan dan Saran

   Langkah terakhir dalam penelitian ini adalah menyimpulkan hasil-hasil yang dicapai dalam penelitian dan memberikan saran

1.7.1 Alur Penelitian

Berikut ini adalah alur tahapan penelitian yang digunakan, seperti ditunjukan pada gambar 1.2.

1.8 Sistematika Penulisan

Penelitian tugas akhir ini disusun dalam beberapa bab dengan tujuan memudahkan alur proses berpikir, dengan sistematika sebagai berikut:

- Bab I Pendahuluan.

  Bab ini menjelaskan latar belakang masalah, perumusan masalah, tujuan penelitian, ruang lingkup pembahasan, batasan masalah, metodologi penelitian dan sistematika penulisan.

- Bab II Tinjauan Teori.

  Bab ini menjelaskan teori-teori terkait dan penelitian-penelitian sebelumnya yang berhubungan dengan penelitian ini. Sebagai dasar teori yang akan digunakan didalam proses pencarian solusi dari permasalahan yang menjadi objek pada penelitian ini.

- Bab III Pengembangan Model.

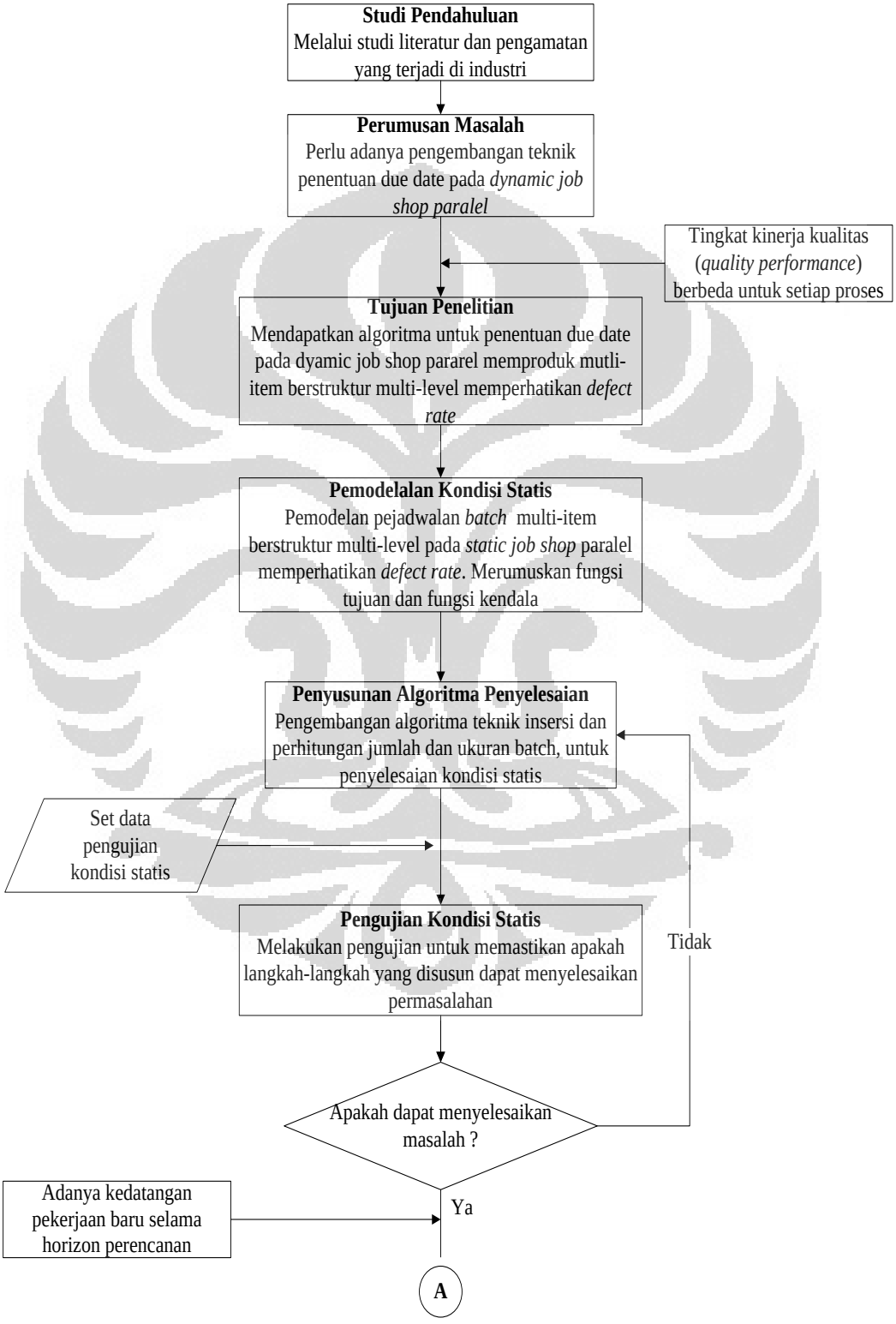
  Bab ini menjelaskan pengembangan model satu dan model dua. Dimana model satu pengembangan model untuk kondisi pemenuhan *due date*, sedangkan pada model dua pengembangan model untuk penentuan *due date*. Algoritma pencarian solusi pada kedua model, menggunakan pendekatan teknik penyisipan dengan ukuran performansi *total actual flow time.*

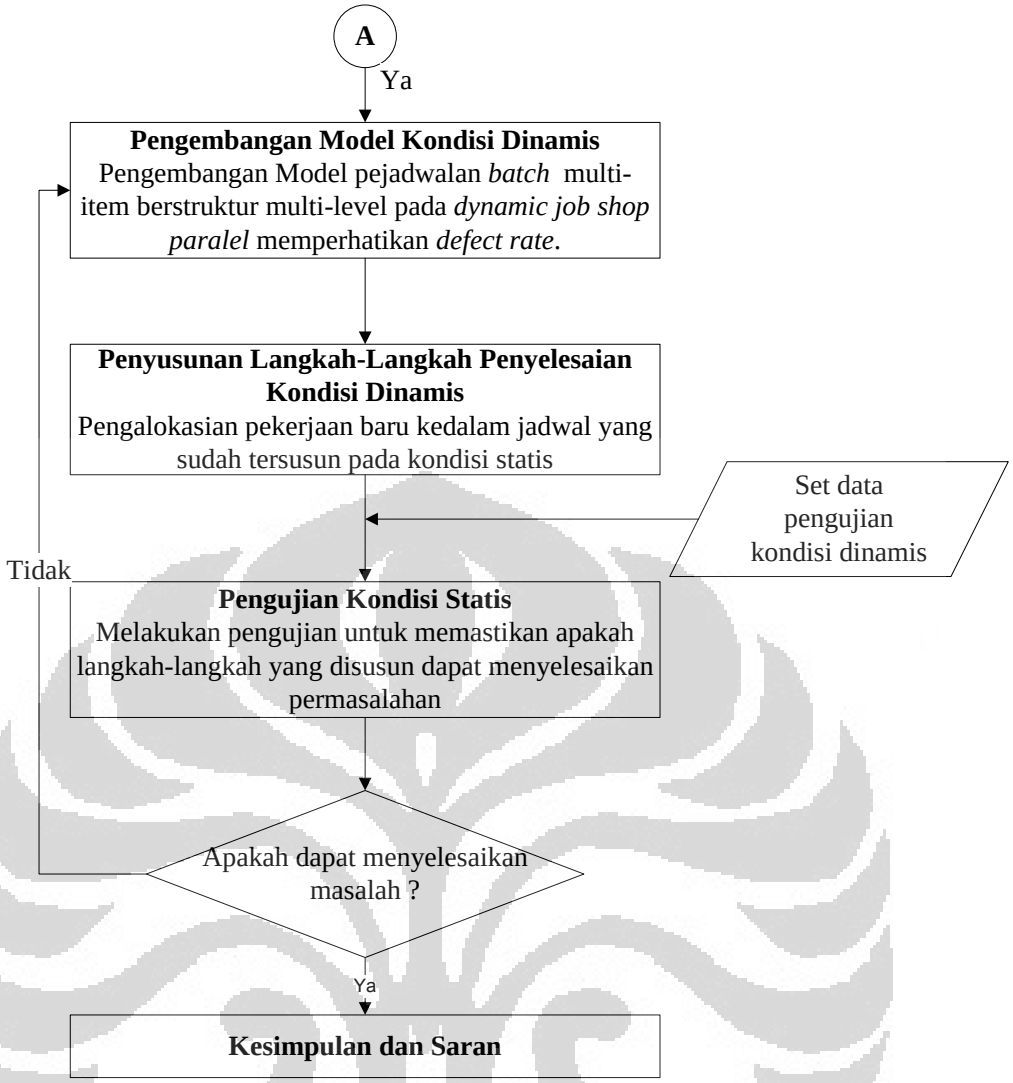
- Bab IV Pengujian dan Analisa Model.

  Bab ini menjelaskan proses verifikasi dan validasi dari model yang dikembangkan, dengan menggunakan hipotesis data. Proses ini, dilakukan untuk memastikan algoritma yang dikembangkan dapat bekerja dengan baik, untuk menyelesaikan permasalahan.

- Bab V Kesimpulan dan Saran.

  Bab ini berisi kesimpulan dari hasil penelitian yang dilakukan dan saran-saran untuk penelitian selanjutnya yang dapat dikembangkan dari penelitian ini.





Gambar 1. 2 Diagram Alur Penelitian

BAB II
TINJAUAN TEORI

2.1 Lingkungan Produksi *Just In Time* (JIT)

Sistem *Just In Time* (JIT) dikembangkan di Toyota Motor Company di Jepang. Meskipun Schonberger (1982) mengisyaratkan bahwa JIT mungkin telah ada sejak 20 tahun lalu atau lebih dalam industri galangan kapal Jepang. Namun, aplikasi modern dari JIT dipopulerkan pada pertengahan tahun 1970-an di Toyota oleh Mr Taiichi Ohno, wakil presiden Toyota dan beberapa kologenya. Selanjutnya konsep ini mulai diadopsi oleh industri-industri di Amerika serikat.

Sistem JIT memiliki tujuan dasar untuk meningkatkan laba dan pengembalian investasi melalui pengurangan persediaan dan peningkatan kualitas. Tujuan ini dapat dicapai dengan menghilangkan berbagai pemborosan dan perencanaan produksi yang sesuai dengan kebutuhan dalam arti tepat dari segi kuantitas dan kualitas. Sistem ini mengintegrasikan permasalahan persediaan dan pejadwalan dalam satu pertimbangan dengan tujuan meminimumkan tingkat persediaan yang dipandang sebagai pemborosan, sekaligus memenuhi kebutuhan konsumen atau pelanggan yang dinyatakan dalam batas waktu pemenuhan order (*due-date*).

Dalam penelitian ini, model penjadwalan yang diusulkan diarahkan pada lingkungan JIT, sehingga tidak diperkenankan terjadinya keterlambatan penyelesaian pekerjaan, serta pekerjaan yang sudah selesai akan segera langsung dikirimkan ke pelanggan sesuai dengan batas waktu yang diinginkan oleh pelanggan, pada sistem produksi JIT. Dengan mempertimbangkan kinerja kualitas pada setiap tahapan proses yang dilalui, karena faktor ini memiliki pengaruh yang cukup signifikan dalam pemenuhan *due-date*.

2.2 Konsep Penjadwalan

Penjadwalan adalah suatu proses untuk memutuskan bagaimana menjalankan berbagai sumber daya pada berbagai tugas yang mungkin. Waktu dapat ditentukan atau dibuat mengambang sebagai bagian dari urutan kejadian.Teori penjadwalan fokus pada pemodelan matematis yang berhubungan dengan proses penjadwalan. Pengembangan model mengarah pada teknik pencarian solusi dan permasalahan secara praktikal, yang berlanjut dengan hubungan antara teori dan

praktek, Baker dan Trietsch (2009). Pemodelan yang dilakukan bertujuan untuk mencakup semua struktur masalah kedalam bentuk matematis.

Terdapat tiga jenis pengambilan keputusan yang lazim dilakukan yaitu *turnaround, timeliness* dan *throughput*. *Turnaround* mengukur waktu yang dibutuhkan untuk menyelesaikan pekerjaan yang tersedia. *Timeliness* mengukur kesesuaian penyelesaian pekerjaan tertentu selama tenggang waktu yang diberikan. Sedangkan *throunghput* mengukur jumlah pekerjaan yang telah diselesaikan selama periode waktu yang telah ditetapkan (*due date).* Penjadwalan diperlukan agar alokasi tenaga operator, mesin dan peralatan produksi, urutan proses, jenis produk, pembelian material dan sebagainya menjadi efisien. Di samping keputusan perencanaan jangka menengah yang tanpa memerhatikan urutan kegiatan produksi, ada masalah lain yang disebut penjadwalan yang mana alokasi sumberdaya dan urutan pengerjaan menjadi sangat penting.

2.2.1 Pendekatan Penjadwalan

Terdapat dua pendekatan dasar dalam penjadwalan yang digunakan untuk menyusun suatu jadwal yaitu pendekatan maju *(forward approach*) dan pendekatan mundur (*backward approach*). Pendekatan maju adalah pengurutan pekerjaan yang bertolak dari arah sekarang (waktu sama dengan nol) dan bergerak menuju ke waktu yang akan datang, sedangkan penjadwalan mundur adalah penjadwalan yang dimulai dari *due-date* ke arah waktu nol. Pada pendekatan maju akan dihasilkan suatu jadwal yang layak, tetapi tidak menjamin *due-date* akan terpenuhi, sementara dengan pendekatan mundur akan diperoleh jadwal yang memenuhi *due-date*, tetapi tidak ada jaminan jadwal yang diperoleh tersebut layak. Pada kondisi keterlambatan penyelesaian pekerjaan tidak diijinkan, pendekatan yang paling tepat digunakan adalah penjadwalan dengan menggunakan pendekatan mundur (*backward approach).* Sedangkan pada permasalahan penentuan *due date* maka pendekatan yang paling tepat digunakan adalah pendekatan maju *(forward).* Agar diperoleh waktu penyelesaian pekerjaan yang dimulai dari waktu nol sebagai *due date* yang akan diajukan kepada konsumen.

Penjadwalan *batch* dengan pendekatan mundur berarti urutan *batch* atau pekerjaan dijadwalkan secara mundur mulai dari *due-date*. *Batch* pertama terjadwal merupakan *batch* terakhir yang akan diproses sesuai dengan urutan

waktu dan *batch* terakhir yang terjadwal adalah *batch* pertama yang akan diproses. Pendekatan mundur ini sesuai dengan konsep *just-in-time* karena item akan selesai tepat pada saat diperlukan yaitu pada saat *due-date*.

Untuk menyesuaikan dengan kondisi *just-in-time* serta model penjadwalan mundur, Halim dan Ohta (1993), telah mengembangkan kriteria performansi yang disebut dengan *total actual flow time*.

2.2.2 Penjadwalan *Batch*.

Penjadwalan *batch* melakukan pemecahan suatu pekerjaan kedalam kelompok pekerjaan (*group of jobs)*, pemecahan ini dilakukan untuk memperkecil waktu pemroses. Halim (1993) dalam mengemukakan beberapa teorema penjadwalan *batch:*

Teorema 2.1

Bila terdapat *N batch* dari satu tipe item, dengan waktu *set-up* konstan yang diproses pada satu mesin, maka *batch* yang memberikan *total actual flow time*, adalah urutan yang disusun dengan rasio berikut:

$$(tQ_{[1]}+ s)/Q_{[1]} \leq tQ_{[2]}+ s)/Q_{[2]} \leq \ldots\ldots \leq tQ_{[N]}+ s)/Q_{[N]}$$

(2.6)

Bukti:

Andaikan ada dua jadwal layak untuk *N batch*. Jadwal pertama memperlihatkan bahwa *batch* g adalah *batch* pada urutan ke-g dan *batch* (g+1) pada urutan ke-(g+1), kedua diurutkan secara mundur.

Jadwal kedua berbeda dari jadwal pertama, hanya pada kondisi *batch* g berada pada urutan ke-(g+1) dan *batch* (g+1) berada pada urutan ke-g. Misalkan F^{a1} dan F^{a2} masing-masing adalah *total actual flow time* seluruh *part* di *shop* untuk jadwal pertama dan kedua, maka dari persamaan (2.6) dapat ditulis:

$$F^{a1}-F^{a2}=(tQ_{[g]}Q_{[g+1]+}sQ_{[g+1]})- (tQ_{[g]}Q_{[g+1]+}sQ_{[g]})$$

(2.7)

Dengan demikian jadwal pertama akan memberikan *total actual flow time* yang lebih baik (lebih kecil) atau sama dengan jadwal kedua, atau $F^{a1} \leq F^{a2}$, jika dan hanya jika :

$$(tQ_{[g]}Q_{[g+1]+}sQ_{[g+1]}) \leq (tQ_{[g]}Q_{[g+1]+}sQ_{[g]})$$

(2.8)

atau

$$tQ_{[g]}+ s)/Q_{[g]} \leq tQ_{[g+1]}+ s)/Q_{[g+1]}$$

(2.9)

Teorema 2.2

Misalkan terdapat *N batch* yang masing-masing *batch* terdiri dari item tunggal. Jadwal optimal dari *N batch* yang meminimasi *total actual flow time* seluruh *part* dari *multi item* didapat dengan mengurutkan *batch* yang sesuai denga rasio berikut:

$$(t_{[1]}Q_{[1]}+ s_{[1]})/Q_{[1]} \leq t_{[2]}Q_{[2]}+ s_{[2]})/Q_{[2]} \leq \ldots\ldots \leq t_{[N]}Q_{[N]}+ s_{[N]})/Q_{[N]}$$

(2.10)

Bukti:

Prosedur pembuktian sama dengan Teorema 1 hanya *t* dan *s* memakai tambahan indeks yang menyatakan jenis item yang berbeda.

2.2.3 Penjadwalan *Job Shop*

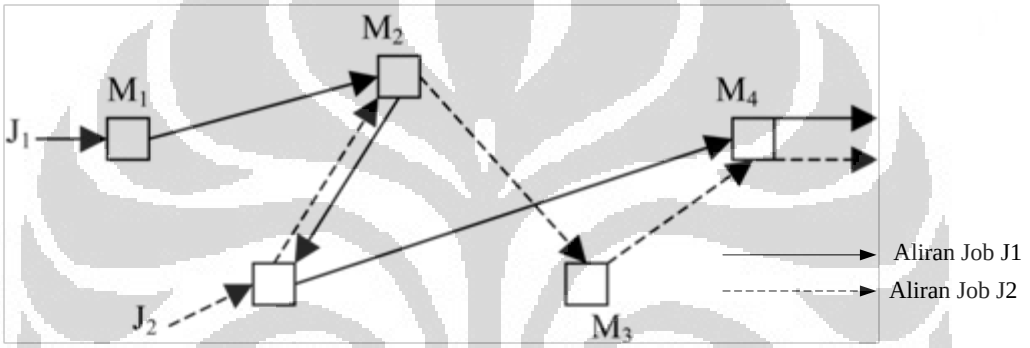
Job shop adalah suatu lingkungan manufaktur dimana *job-job* yang datang memiliki rute pengerjaan atau operasi yang seringkali tidak sama. Bentuk sederhana dari model ini mengasumsikan bahwa setiap *job* hanya melewati satu jenis mesin sebanyak satu kali dalam rutenya pada proses tersebut. Namun ada juga model lainnya dimana setiap *job* diperbolehkan untuk melewati mesin sejenis lebih dari satu kali pada rutenya. Model ini disebut juga *job shop* dengan *recirculation* (pengulangan).

Elemen dari permasalahan *job shop* adanya sejumlah *n* pekerjaan (*J),* j=1,…n, yang harus diproses pada sekumpulan *M* mesin. Pekerjaan J_j terdiridari n_j operasi O_{ij}, i=1,…,n_j, yang harus diproses dalam suatu rute tertentu, O_{ij} operasi ke-*i* dari pekerjaan ke J_j. Di mana masing-masing operasi diproses pada mesin tertentu (*dedicated machine*). Sehingga lingkungan produk *job shop* memiliki karakteristik sebagai berikut:

- Terdapat sejumlah *m* mesin dan *n* pekerjaan.

- Setiap pekerjaan memiliki rute penyelesaian pemprosesan yang berbeda satu sama lain.
- Setiap operasi dalam pekerjaan diproses oleh salah satu mesin yang ada dengan waktu proses yang diasumsikan tetap.
- Setiap proses operasi dapat melewati satu jenis mesin lebih dari satu kali.
- Permasalahan penjadwalan untuk model *job shop* merupakan salah satu permasalahan optimasi kombinatorial yang kompleks sehingga disebut *NP-hard* (*NP* merupakan singkatan dari *nondeterministic polynomial*).

Bentuk permasalahan penjawalan *job shop* dapat digambarkan seperti pada gambar berikut ini.



Gambar 2.1 Model Rute *Job Shop*

Didalam penyusunan jadwal *job shop* diperlukan adanya aturan pengurutan pekerjaan ke dalam mesin-mesin sebagai jadwal awal yang akan dilakukan perbaikan pengurutan pekerjaannya. Penelitian eksperimen telah menunjukan bahwa penyusunan penjadwalan berdasarkan perioritas aturan pengurutan sebagai metode praktis untuk mencari solusi yang baik dalam permasalahan *job shop* meskipun tentunya kondisi optimal tidak dapat dijamin. Terdapat beberapa aturan prioritas umum yang dapat digunakan secara efektif di dalam proses penjadwalan dalam lingkungan *job shop*, yaitu:

- *Shortest Processing Time* (SPT) yaitu menilih waktu operasi dengan waktu proses yang paling minimum.
- *First Come First Served* (FCFS) yaitu memilih operasi yang datang pada mesin lebih awal.
- *Most Work Remaining* (MWKR) yaitu memilih operasi terkait dengan pekerjaan yang paling memiliki pekerjaan tersisa.

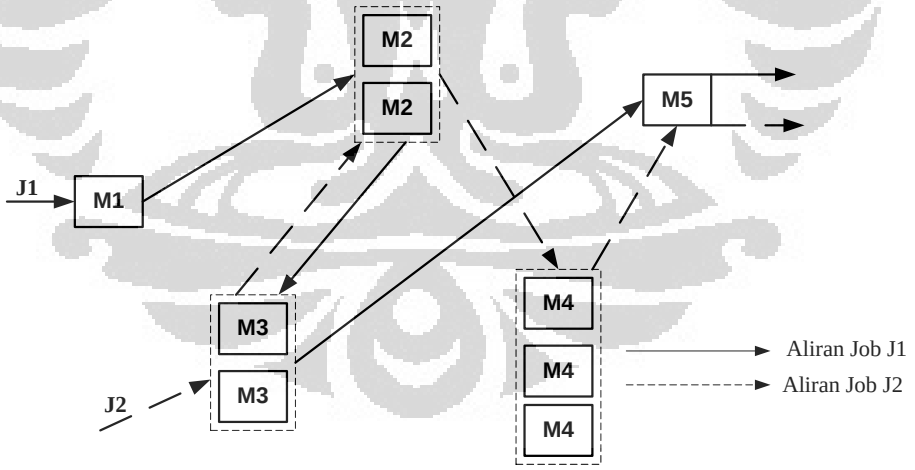
- *Least Work Remaining* (LWKR) yaitu memilih operasi terkait dengan pekerjaan yang memiliki pekerjaan paling sedikit tersisa untuk diproses.

Pada kondisi penjadwalan dengan kriteria meminimasi *total flow time*¸ SPT dan LWKR biasanya lebih efektif dari aturan yang lainnya. Sedangkan untuk kriteria meminimasi *makespan,* maka MWKR dan FCFS memberikan pendekatan yang lebih baik.

2.2.4 Mesin Paralel

Secara umum, penjadwalan membutuhkan keputusan terkait dengan pengurutan pekerjaan dan alokasi sumber daya. Ketika pada kondisi terdapat sejumlah *m* mesin identik yang dapat digunakan memproses *n* pekerjaan secara bersamaan, dan diasumskan bahwa pekerjaan dapat diproses melalui paling tidak pada satu mesin pada suatu waktu tertentu.

Pada kondisi lingkungan *job shop* mesin paralel berarti terdapat sejumlah *m* mesin identik untuk menyelesaikan *n* pekerjaan pada setiap rute proses yang dilalui oleh suatu pekerjaan. kondisi *job shop* dengan paralel mesin dapat digambarkan seperti pada gambar dibawah ini.



Gambar 2.2 Kondisi *Job Shop* Mesin Paralel

Dalam permasalahan mesin paralel dengan kriteria *total completion time* aturan prioritas SPT memberikan pendekatan yang sangat baik. Berdasarkan aturan SPT, pekerjaan dengan waktu pemproses terkecil akan dialokasikan pada mesin 1 kesatu, pekerjaan terkecil berikutnya dialokasi pada mesin 2 kedua, dan seterusnya.

2.3 Konsep dan Definisi *Due date*

Terdapat banyak definisi mengenai *due-date* yang diberikan oleh para pakar *scheduling* diantaranya: didefinisikan sebagai titik waktu (*point of time*) saat pemprosesan pekerjaan harus dapat diselesaikan, Baker (1974). Bedworth dan Bailey (1987) memperluas definisi *due-date* ini sebagai batas akhir yang ditetapkan terhadap suatu pekerjaan dan bila melebihi batas akhir itu dikatakan terjadi keterlambatan yang dikenakan *penalty* atau denda.

Halim et al. (1998) dalam kaitannnya dengan konsumen mendefinisikan *due-date* sebagai waktu penyerahan produk yang dipesan sesuai dengan kesepakatan antara pihak produsen dan konsumen pada awal proses perencanaan. Lebih spesifik lagi, Halim et al.(1999) mendefinisikan *due-date* d_i, sebagai batas waktu dari operasi terakhir item ke-*i* harus sudah selesai dikerjakan.

Penelitian dalam area penentuan *due date* dalam masalah penjadwalan, mulai dirintis oleh Seidman et al.(1981) dan Panwalkar et al.(1982) dalam Shabtay (2010). Pengembangan penentuan *due date* pada awalnya dilakukan melalui pendekatan *due date* yang berlaku umum untuk semua pekerjaan (*common due date)* yang kenal dengan *CON method*.

Penyelesaian pekerjaan yang lebih awal dari *due-date* akan menyebabkan penumpukan persediaan, sedangkan penyelesaian melebihi *due-date* biasanya akan dikenakan *penalty* oleh pemesan. Dalam penelitian ini tidak diperkenankan terjadinya keterlambatan. Pernyataan yang menarik mengenai *due-date* ini adalah kapan saat terakhir suatu pekerjaan harus dimulai agar dapat selesai tepat pada *due-date*.

2.4 *Total Actual Flow Time*

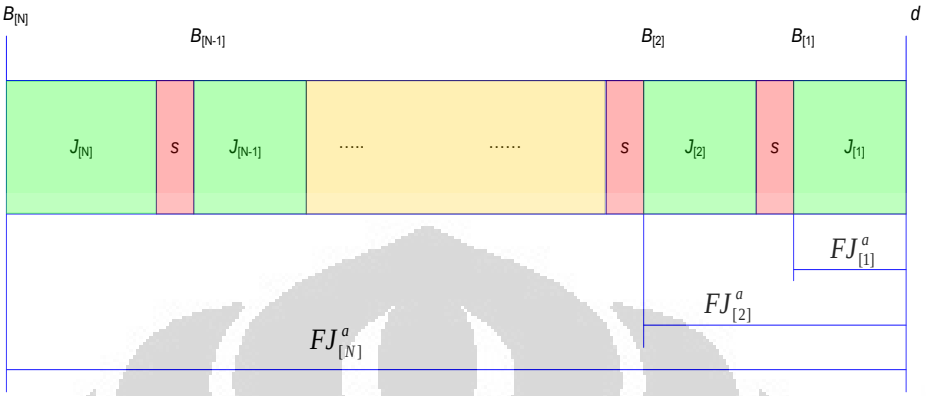
Halim (1993) menyatakan bahwa konsep *actual flow time* sebagai ukuran performansi untuk *shop-time* pertama kali dikemukan Miyazaki dan Ohta (1987), yaitu waktu yang diperlukan suatu pekerjaan tersebut sampai *due-date*, atau

$$FJ_{[i]}^{a} = d - B_{[i]} \; dengan \; i = 1,2, \dots, b \qquad (2.1)$$

Dengan d adalah *common due-date* dan $B_{[i]}$ adalah saat mulai pemprosesan pekerjaan $J_{[i]}$. Selanjutnya, Halim (1993) menurunkan formulasi diatas menjadi sebagai berikut: jika diasumsikan waktu *set-up*, *s*, konstan dan

tidak termasuk ke dalam waktu proses pekerjaan, p_i, maka berdasarkan Gambar 2.1 dapat dituliskan :

$$FJ^a_{[i]} = \{\sum_{j=1}^{i}(p_j + s)\}\text{-s dengan } i = 1,2,\ldots b \qquad (2.2)$$



Gambar 2.3 Waktu Tinggal Aktual (*Actual Flow Time)* Pekerjaan

Bila diketahui ukuran masing-masing *batch*, maka persamaan (2.2) dapat diformulasikan sebagai berikut. Misalkan terdapat N *batch* dari satu jenis item dengan kuantitas masing-masing *batch* $Q_{[i]}$, i = 1,2,….,N, waktu proses satuan tiap part adalah *t*, dan waktu *set-up* antar *batch* adalah *s*. Jika diasumsikan bahwa *batch* terjadwal datang tepat pada saat proses akan dimulai ($B_{[i]}$), dan semua *part* (dari seluruh *batch*) yang selesai diserahkan sekaligus pada suatu *common due-date*, maka dengan pendekatan mundur *actual flow time* untuk *batch* dapat ditentukan melalui persamaan (2.2) dengan mengganti waktu proses pekerjaan dengan waktu proses *batch* yang didapat dari perkalian antara ukuran *batch* dengan waktu proses sebuah *part* , atau

$$FL^a_{[i]} = \{\sum_{j=1}^{i}(tQ_{[j]} + s)\} - s \qquad (2.3)$$

Bila semua *part* dalam suatu *batch* datang pada saat yang bersamaan, maka setiap part tersebut akan tinggal di *shop* sepanjang $FL^a_{[i]}$. Dengan kata lain *actual flow time* masing-masing part sama dengan waktu tinggal aktual *batch*. Jadi, *actual flow time* $FL^a_{[i]}$ untuk selutuh part dalam *batch* $L_{[i]}$ dapat dihitung dengan mengalikan waktu tinggal aktual *batch* dengan jumlah *part* dalam *batch* tersebut, sehingga :

$$F^a_{[i]} = (\{\sum_{j=1}^{i}(tQ_{[j]} + s)\} - s_i).\, Q_{[i]} \; dengan \; i = 1,2,\ldots,N \qquad (2.4)$$

Dengan demikian , maka *total actual flow time* untuk seluruh *part* di dalam *shop*:

$$F^a = \sum_{i=1}^{N}\left(\left\{\sum_{j=1}^{i}\left(tQ_{[j]} + s\right)\right\} - s_i\right).Q_{[i]} \quad (2.5)$$

2.5 Teknik Penyisipan (*Insertion Technique*)

Aplikasi teknik-teknik penyisipan (*insertion techniques*) untuk permasalahan-permasalahan *job shop* dengan waktu *set-up* di bahas oleh Sotkov et al.(1999), di mana permasalahan *job shop* dengan waktu *set-up* diformulasikan sebagai berikut: sejumlah n pekerjaan dipartisikan menjadi *g* group G_1, G_2,….,G_g. Untuk tiap pekerjaan *i*, suatu *routing* teknologikal q^i=(j_1,j_2,…,j_{ki}) diberikan, di mana k_i menyatakan jumlah operasi dari pekerjaan *i* dan j_h dengan $1 \leq j_h \leq m$ adalah mesin ke-*h* di mana pekerjaan *i* harus diproses. Notasi t_{ij} untuk waktu proses operasi (*i,j*) dari pekerjaan *i* pada mesin *j* dan waktu *set-up* $s_{hj} \geq 0$ untuk group G_h pada mesin *j*.

Dalam penelitian diatas, juga dibahas dua jenis permasalahan ketersediaan pekerjaan (*job availability*), yaitu ketersediaan item (*item avaialability*) dan ketersediaan *batch (batch availability)*. Untuk jenis permasalahan pertama yang dipertimbangkan, *set-up* terjadi pada suatu mesin sebelum pekerjaan pertama mulai diperoses dan ketika suatu pekerjaan harus diproses setelah suatu pekerjaan dari group yang berlainan.

Untuk jenis permasalahan kedua, pekerjaaan diproses dalam ukuran *batch*. Suatu *batch* terdiri dari satu atau lebih pekerjaan dari group yang sama, yang diproses secara bersamaan pada suatu mesin. Pemprosesan suatu *batch* pada suatu mesin tertentu dapat dimulai ketika:

- Pemprosesan *batch* pendahulu (*preceding batch*) pada mesin ini telah diselesaikan dan *set-up* telah dilakukan, dan
- Pemprosesan seluruh pekerjaan dari *batch* bersangkutan pada mesin pendahulunya telah diselesaikan.

Waktu proses (*processing time)* suatu *batch* merupakan penjumlahan waktu proses operasi-operasi pada *batch*. Saat selesai tiap operasi pada *batch* adalah sama dengan saat selesai untuk *batch* pekerjaan dari group G_r diproses pada mesin *j*. Namun, dapat pula terjadi kasus dimana ketersediaan *batch* hanya pada suatu *subset* mesin. Notasi (m_1,…m_h) mengindikasikan bahwa ketersediaan *batch* pada mesin-mesin m_1,…m_h,sementara pada mesin-mesin lainnya diasumsikan terdapat ketersediaan item.

Di asumsikan bahwa matrik *Ranking A* untuk suatu jadwal parsial telah ditentukan. Berdasarkan urutan penyisipan yang telah dipilih, operasi *(i,j)* harus dipenyisipankan segera pada semua posisi yang mungkin di mesin *j*. Matriks hasil setelah penyisipanan operasi *(i,j)* ini haruslah berupa matriks *ranking*, semua *routing* dipenuhi, dan semua hubungan presedensi yang sebelumnya telah ada diantara operasi-operasi tertentu tetap berlaku.

Di asumsikan pula bahwa pada mesin *j*, sekumpulan operasi telah diurutkan. Anggota kolom *j* pada matriks *A* adalah $k_1,k_2\ldots,k_u$ dengan $k_1 \leq k_2 \leq \ldots \leq k_u$ dan diasumsikan bahawa $a_{ih,j}=k_h$ untuk $i \leq h \leq u$, sehingga penyisipanan berikut harus dipertimbangkan:

1. Proses pekerjaan *i* sebagai pekerjaan pertama pada mesin *j*: tentukan a_{ij}= max{a_{iv}/operasi*(i,j)*+1}, modifikasi anggota lainnya pada kolom *j* (operasi yang sudah diurutkan pada mesin *j*) dan anggota matriks yang berhubungan dengan semua operasi yang telah diurutkan di mesin *j* tersebut. Jika modfikasi ini mengakibatkan modifikasi pula pada a_{ij}, maka pengpenyisipanan operasi *(i,j)* yang telah dilakukan tidak layak.
2. Proses pekerjaaan *i* sebagai pekerjaan ke-*l* pada mesin *j* dengan ($2 \leq l \leq u+1$): tentukan $a_{ij} = max\ \{max\{a_{iv}$/operasi*(i,v)* mendahului operasi*(i,j)*;k_{l-1}}+1}. Kemudian kelayakan jadwal parsial diuji analog dengan cara yang sama disebutkan diatas. Sehingga, semua operasi yang dimiliki kolom *j* memiliki anggota yang lebih besar dari k_l dan operasi yang dipenyisipankan. Jika hal ini menyebabkan terjadinya siklus pada *digraph*, maka pengpenyisipanan tersebut tidak layak.

Algoritma Sotkov et al (1999) dapat disusun sebagai berikut:

Langkah 1. Indentifikasi *routing*, kelompok, waktu *set-up*, waktu operasi, saat datang, *due-date*, dan bobot masing-masing pekerjaan.

Langkah 2. Tentukan saat yang paling awal mulai operasi (*R'*) untuk semua operasi dari masing-masing pekerjaan. Untuk operasi pertama suatu pekerjaan: *R'*= max {saat datang, waktu *set-up* }, dan untuk operasi yang bulan pertama : *R'*=saat selesai paling cepat dari operasi sebelumnya.

Langkah 3. Pilih pekerjaan dengan bobot terbesar. Jadwalkan secara mundur operasi terakhir pekerjaan tersebut (operasi yang terkahir

dilakukan dalam pemprosesan pekerjaan) agar selesai tepat pada saat *due-date*.

Langkah 4. Set j =1.

Langkah 5. Jadwalkan secara mundur semua operasi lainnya dari pekerjaan pada langkah 3 (operasi-operasi pendahulu) sesuai dengan *routing*.

Langkah 6. Pilih pekerjaan dengan bobot terbesar berikutnya. Jadwalkan operasi terakhir pekerjaan tersebut agar selesai tepat pada saat *due-date*. Bila tidak mungkin selesai pada saat *due-date* :

a). Jadwalkan secara mundur operasi tersebut sebelum operasi yang sudah terjadwal, hitung ongkos yang timbul;

b). Jadwalkan operasi tersebut secara mundur setelah operasi yang sudah terjadwal, hitung ongkos yang timbul,dan

c). Penyisipankan operasi tersebut secara mundur diantara operasi yang sudah terjadwal sedekat mungkin dengan *due-date*, hitung ongkos yang timbul. Pilih jadwal dengan ongkos terkecil sebagai dasar penjadwalan selanjutnya. Ongkos $\sum w_i (C_i - d_i)^2$.

Langkah 7. Set $j=j+1$.

Langkah 8. Jadwalkan operasi lainnya. Periksa apakah saat mulai $\geq$ saat paling awal mulai operasi ? Bila ya, lanjutkan ke *Langkah 9* dan bila tidak, geser jadwal ke kanan sebanyak waktu yang diperlukan agar pekerjaan yang sedang dijadwalkan layak. Hitung ongkos yang timbul.

Langkah 9. Ulangi *Langkah 6-8*. Lanjut ke *Langkah 10.*

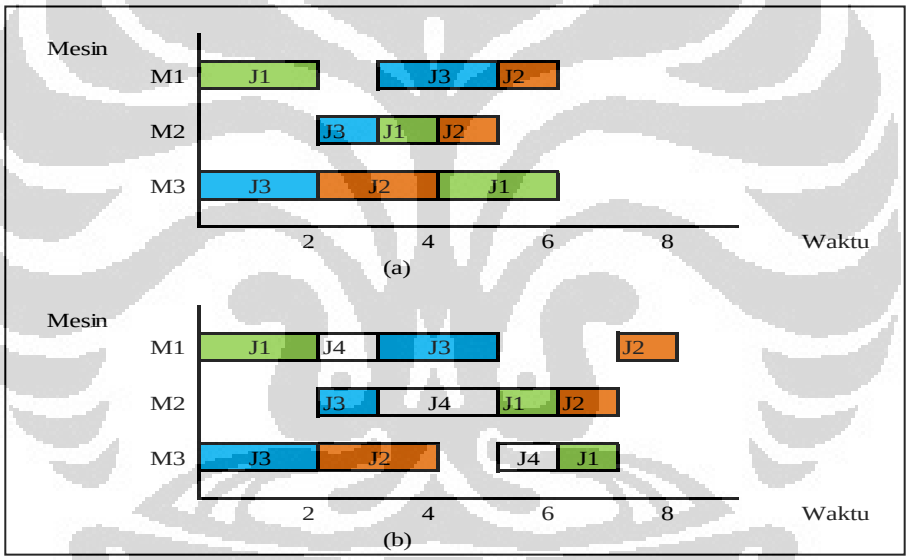
*Langkah 10.*Apakah j sudah sama dengan jumlah pekerjaan ? Bila ya, penjadwalan selesai dan lanjutkan ke *Langkah 11*, dan bila tidak, ulangi *Langkah 9.*

Langkah 11. Periksa apakah ada alternatif jadwal dengan ongkos yang lebih kecil. Perbaikan dilakukan dengan menggeser jadwal lebih dekat kearah *due-date* nya.

Langkah 12. Pilih alternatif jadwal dengan ongkos terkecil sebagai jadwal definitif.

Pengembangan teknik penyisipan pada kondisi statis dan dinamis didalam menyelesaikan permasalahan dengan keterbatasan sumber daya telah dilakukan oleh Artigues et al(2003). Dari hasil penelitian ini menunjukan bahwa algoritma teknik penyisipan memberikan penjadwalan yang lebih baik pada kondisi lingkungan dinamis, pengujian dilakukan di dalam penyelesaian permasalahan penjadwalan proyek.

Penelitian selanjutnya, dilakukan oleh Groflin et al (2006), melakukan kajian kelayakan penyisipan atau penyisipkan pekeerjaan pada *multi processor task job shop* yang bertujuan utuk meminimasi *makespan*. Dari hasil penelitian ini terlihat bahwa teknik penyisipan memberikan waktu pencairan yang lebih pendek, jika dilihat dari CPU time. Konsep dasar dari teknik penyisipan dapat dilihat pada gambar 2.4.



Gambar 2.4 Konsep Dasar Teknik Penyisipan

2.6 Penjadwalan Ulang (*Rescheduling*)

Penjadwalan ulang (*rescheduling)* merupakan suatu proses interatif atau berulang dari tiga langkah berikut Wu dan Li (1995):

1) Langkah evaluasi, yaitu mengevaluasi dampak atau pengaruh terjadinya faltor-faktor *rescheduling*. Jika dampak bisa diterima, misalnya faktor-faktor *rescheduling* tidak mempengaruhi jadwal yang ada, maka tidak dibutuhkan tindak lanjut apapun.

2) Langkah solusi, yaitu menentukan solusi *rescheduling* yang dapat meningkatkan performansi jadwal yang ada. Akan tetapi, cara menentukan solusi *rescheduling* masih terbuka sebagai masalah penelitian dan konsekuensinya adalah langkah ini merupakan bagian yang paling sulit dalam *rescheduling*.
3) Langkah perbaikan atau revisi, yaitu mirip dengan langkah evaluasi kecuali inputnya yang berbeda yaitu inputnya adalah solusi *rescheduling*. Jadwal yang ada direvisi dan diperbaharui. Jika hasil jadwal yang direvisi dapat diterima, maka selesai. Jika tidak, maka ulangi langkah 2 dan 3 sampai diperoleh jadwal revisi yang dapat diterima.

Faktor-faktor yang menimbulkan gangguan pada jadwal semula disebut dengan faktor-faktor penjadwalan ulang (*rescheduling factors)*, diantaranya adalah Li et al. (1993) kerusakan mesin, kedatangan pesanan baru yang harus segera dikerjakan, kekurangan material, masalah kualitas, perkiraan waktu proses yang terlalu tinggi atau rendah, pembatalan pesanan, dan perubahan *due-date* bisa maju atau mundur.

BAB III
PENGEMBANGAN MODEL

3.1 Skenario Pengembangan Model

Penelitian ini mengembangkan dua buah model algoritma yang terdiri dari model satu pemenuhan *due date* dan model dua penentua *due date*, masing-masing model dikembangkan untuk kondisi dinamis. Pengembangan model yang dilakukan terlebih dahulu diawali dengan pengembangan model pada kondisi statis, pada kondisi ini diasumsikan semua item (*job*) yang harus dikerjakan tersedia pada awal periode perencanaan (*t=0*). Setelah permasalahan pada kondisi statis dapat terselesaikan, maka dilakukan pengembangan pada kondisi dinamis, didalam permasalahan terjadi kedatangan item baru yang harus dikerjakan disepanjang horizon perencanaan.

Pada kondisi dinamis, pada prinsipnya akan melakukan evaluasi dan melakukan pencarian solusi atas kedatangan item-item baru yang terjadi disepanjang horizon perencanaan. Dari solusi kondisi dinamis ini, akan menunjukan item baru mana saja yang realistis untuk dikerjakan, dengan memperhatikan item-item yang sudah terjadwal sebelumnya. Kedua model yang dikembangkan bertujuan untuk meminimasi *total actual flow time* dengan menggunakan teknik penyisipan, didalam proses pencarian solusi.

3.2 Notasi dan Definisi

Dalam pemodelan permasalahan penjadwalan *batch* pada sistem produksi *job shop* dinamis mesin paralel yang memproduksi multi item berstruktur multi level, didalam pemenuhan dan penentuan *due date*. Kondisi permasalahan yang dibahas seperti dijelaskan didalam **Sub Bab 3.2**, digunakan notasi-notasi sebagai berikut.

a. Subcript

i = menyatakan jenis item i yang akan diproduksi, sehingga $i=1,2,\ldots r$. dimana r menyatakan banyaknya item yang akan diproduksi

j = menyatakan komponen ke-j dari c_{i0} jenis komponen untuk membuat item i, sehingga $j=1,2,\ldots c_{i0.}$

k = menyatakan urutan ke k dari h_{ij} proses yang harus dilalui didalam pembuatan komponen ke-j dari item i, sehingga $k=1,2,\ldots h_{ij}$

l = menyatakan level ke-l dari e_{i0} dari item i, sehingga $l=0,1,2,\ldots e_{i0}$.

m = menyatakan mesin ke-m dari v mesin yang tersedia, sehingga $m=1,2,3\ldots v$.

n = menyatakan mesin identik ke-n dari jumlah mesin w pada setiap tahapan proses atau kelompok mesin, sehingga $n=1,2,3\ldots w$.

u = menyatakan batch ke- u dari *N batch* untuk setiap komponen p_{ij}

b. Parameter

c_{i0} = menyatakan jumlah jenis komponen yang diperlukan untuk membuat item i, dimana c_{i0} tidak menyatakan jumlah level.

h_{ij} = menyatakan jumlah proses yang harus dilalui didalam pembuatan komponen ke j dari item ke i.

e_{i0} = menyatakan jumlah level yang dimiliki oleh item i didalam struktur produknya.

v = menyatakan jumlah jenis mesin yang tersedia didalam sistem produksi.

w = menyatakan jumlah jenis mesin identik yang tersedia untuk setiap mesin ke-m.

p_{i0} = menyatakan item i yang akan diproduksi; $i=1,\ldots,r$.

p_{ij} = menyatakan komponen ke-j dari item *i*.

n_{ij} menyatakan jumlah komponen ke-*j* dari item *i.*

$Z(p_{ij})$ = set dari induk-induk komponen j dari item i berdasarkan struktur produknya.

H_{ij} = banyaknya komponen j yang dibutuhkan untuk membuat satu unit induk langsung didalam struktur produknya (*bill of material),* dapat juga diartikan sebagai jumlah komponen *j* yang diperlukan untuk membuat komponen pada level diatasnya.

d_{i0} = *due date* item i.

$O_{i0k0m\bullet}$ = operasi ke-k untuk item i pada level ke-*0* yaitu p_{i0} yang dilakukan di mesin m untuk semua *n*; m=1,2...,v, *n=1,2…w*

O_{ijklmn} = operasi ke-k untuk komponen p_{ij} yang berada pada level ke-*l* yang dilakukan di mesin m ke *n*. m=1,2...,v, *n=1,2…w*

$t_{i0km\bullet}$ = waktu proses operasi $O_{i0klm\bullet}$ untuk setiap unit p_{i0}.

$t_{ijkm\bullet}$ = waktu proses operasi $O_{ijklm\bullet}$ untuk setiap unit p_{ij}.

$s_{ijkm\bullet}$ = waktu set-up semua mesin m identik untuk operasi $O_{ijklm\bullet}$.

$dr_{ijklm\bullet}$ = tingkat cacat (*defect* rate) operasi ke-*k* level ke-*l* dari komponen p_{ij} pada semua mesin *m.*

- Variabel

TF = *total actual flow time.*

Q_{i0} variabel yang menyatakan unit item p_{i0} yang dibuat.

Q_{ij} = variabel yang menyatakan unit komponen p_{ij} yang akan dibuat.

S_{i0klmn} = variabel yang menyatakan saat dimulainya operasi O_{i0k0mn}

S_{ijklmn} = variabel yang menyatakan saat dimulainya operasi O_{ijklmn}.

C_{i0k0mn} = variabel yang menyatakan saat selesainya operasi O_{i0k0mn}.

C_{ijklmn} = variabel yang menyatakan saat selesainya operasi O_{ijklmn}.

N_{ijk} = jumlah *batch* untuk operasi ke-k dari komponen p_{ij} pada operasi O_{ijklmn}.

$L_{ijkmn}[b]$ = *Batch* untuk operasi ke-k dari komponen p_{ij} pada operasi O_{ijklmn} yang berada di posisi b, diproses di mesin m ke-n.

$Q_{ijkm}[b]$ = jumlah *part* dalam *batch* $L_{ijkmn}[b]$

$B_{ijklmn}[N]$ = menyatakan waktu mulai diprosesnya *batch* ke *N* dari item p_{ij} untuk level ke-*l* operasi ke-*k* pada mesin *m* ke-*n*.

3.3 Model Penjadwalan Pemenuhan *Due Date*

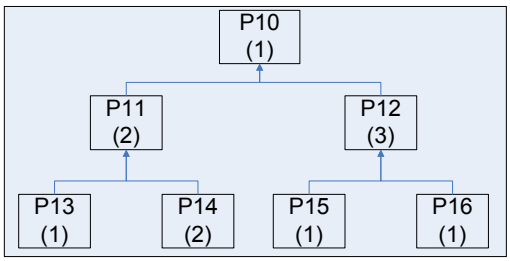
Sistem yang dibahas pada model penjadwalan ini adalah sistem yang memproduksi beberapa jenis item dan berstruktur multi level. Dalam sistem ini setiap item memiliki struktur produknya (*Bill of Material*-BoM) masing-masing, dimana produk akhir dan setiap komponen pada tiap levelnya dimungkinkan untuk diproses dibeberapa mesin berbeda yang juga dengan *routing* (urutan) yang berbeda-beda.

Sistem produksi yang ditinjau berlingkungan *Just In Time* (JIT), dengan memperhatikan tingkat cacat pada setiap operasi yang dilalui selama proses produksi. Kondisi ini selaras dengan lingkungan JIT, dalam JIT kualitas dari produk yang dihasilkan harus sesuai dengan spesifikasi konsumen atau dengan

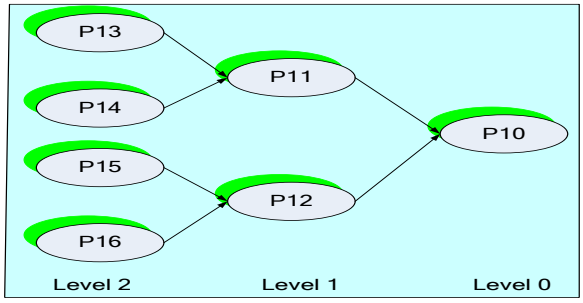
kata lain kualitas harus tetap dijaga. Masalah tersebut dapat diformulasikan sebagai berikut.

Terdapat r jenis item, yang dinyatakan dalam p_{i0}, dengan indeks $i = 1,2,...,r$. Di mana setiap p_{i0} akan diproduksi sebanyak n_{i0} yang harus diselesaika pada *due-date*, d_{i0}. Tiap item yang diproduksi memiliki level sebanyak e_{i0} *level* pada struktur produknya, sehingga $l = 0,1,2,...,e_{i0}$ dan diperlukan sebanyak c_{i0} jenis komponen, dengan setiap jenis komponen dinyatakan dalam p_{ij}, sehingga $j = 0,1,2,...,c_{i0}$. Dengan jumlah item p_{ij} yang akan diproduksi sebanyak n_{ij}. Setiap item dan komponennya diproses dalam h_{ij} operasi, dengan urutan operasi dinyatakan dalam *k* sehingga $k = 1,2,...,h_{ij}$, pada suatu sistem produksi *job shop* yang mempunyai *v* jenis mesin dengan indeks $m = 1,2,...,v$ dan *w* mesin identik dimana mesin identik ke-1, ke-2 dan seterusnya dinyatakan dalam *n*, sehingga $n = 1,2,3...,w$.

Setiap mesin memiliki waktu *set-up* s_{ijkmn} dan waktu proses t_{ijkmn}, dimana waktu operasi sama untuk semua *n* mesin identik. Dengan masing-masing proses operasi memiliki tingkat cacat untuk setiap item dan komponen dinyatakan dalam dr_{ijklmn}. Dalam hal ini, besarnya tingkat cacat sama untuk semua mesin paralel yang identik pada setiap tahapan pemprosesan. Kedatangan pekerjaan bersifat dinamis yang artinya tidak harus semua pekerjaan tersedia pada awal horizon perencanaan, tetapi terjadi kedatangan pekerjaan-pekerjaan disepanjang horizin perencanaan. Kondisi multi level beserta *directed graph* (*digraph*) dapat digambarkan seperti pada gambar 3.1, berikut ini.



Gambar 3.1 Item Berstruktur Multi-Level



Gambar 3.2 Digraph dari Produk Akhir dan Komponennya

Gambar 3.1 dan 3.2 menjelaskan bahwa untuk membuat item p_{10} diperlukan sebanyak enam buah komponen yang terdiri yang terdiri dari tiga level, dimulai dari level nol sampai dengan level dua ($e_{10}=3$). Didalam membuat satu item p_{10} diperlukan komponen p_{11} sebanyak dua unit (H_{11}=2 unit) dan komponen p_{12} sebanyak tiga unit (H_{12}=3 unit). Sehingga item p_{10} baru dapat diproses setelah komonen p_{11} dan p_{12} selesai diproses, karena p_{10} adalah induk langsung dari komponen p_{11} dan p_{12}.

Model yang dikembangkan ini harus dapat menjawab permasalahan sebagai berikut;

- Bagaimana menentukan jumlah *batch* N_{ij}dan ukuran batch $Q_{ij[u]}$ dengan u=1,2,…. N_{ij} untuk setiap item dan komponen-komponenya berdasarkan tingkat cacat untuk setiap operasi dan menjadwalkannya pada setiap mesin dengan memperhatikan ketersediaan mesin.
- Bagaimana melakukan penjadwalan ulang bila terjadi kedatangan pesananan-pesanan baru.

Dalam pengembangan model penjadwalan *batch* pada *job shop* dinamis dengan menggunakan teknik penyisipan untuk kondisi multi item berstruktur multi level diawali dengan pengembangan model pada kondisi statis, kemudian dilanjutkan pengembangan algoritma untuk kondisi dinamis. Permasalahan penjadwalan tersebut diformulasikan kedalam fungsi tujuan dan kendala yang harus dipenuhi dalam proses pencarian solusi.

Dengan melakukan modifikasi pada persamaan (2.1), maka diperoleh fungsi tujuan,

$$\text{Minimasi } TF = \sum_{i=1}^{r}\left(d_{i0} - B_{i\bullet 1 e\bullet\bullet[N]}\right) \qquad (3.1)$$

Persamaan (3.1) menjelasakan fungsi tujuan minimasi total waktu tinggal aktual untuk semua item, dengan $B_{i\bullet 1e\bullet\bullet}[N]$ menyatakan saat mulai *batch* terakhir untuk operasi pertama dari komponen yang berada pada *level e* struktur produk item i yang terjadwal paling akhir, dari sejumlah *N batch*. Dimana $B_{ijklmn}[N]$ menyatakan waktu mulai diprosesnya *batch* ke *N* dari item p_{ij} untuk level ke-*l* operasi ke-*k* pada mesin *m* ke-*n*.

Adapun fungsi-fungsi kendala dari permamasalahan ini terdari beberapa kendala yang berhubungan dengan ketersediaan mesin dan hubungan ketergantungan pekerjaan pada setiap levelnya.

- Fungsi kendala hubungan kuantitas item dengan kuantitas komponen.

$$n_{ij} = n_{i0} * H_{ij}\left(\prod_{p_{iq} \in Z(p_{ij})} H_{iq} \right), \quad H_{iq} \geq 1 \; ; \forall i, j \qquad (3.2)$$

  Persamaan (3.2) menyatakan hubungan kuantitas item yang dibuat dengan kuantitas komponen-komponen penyusunnya. H_{ij} menyatakan jumlah komponen ke-*j* yang diperlukan untuk membuat item *i*, untuk membuat satu unit induk langsung. Dengan p_{iq} menyatakan komponen ke-*q* sebagai elemen himpunan $Z(p_{ij})$ dari induk-induk komponen ke-*j* hingga item *i*. persamaan ini untuk menjamin bahwa didalam penentuan jumlah komponen ke-*j* harus memperhatikan struktur produk penyusunannya.

- Fungsi kendala hubungan tingkat cacat (*defect rate*) terhadap jumlah komponen yang diproduksi.

$$n_{ijk(diproduksi)} \geq n_{ij(k+1)(diproduksi)} * (1 + dr_{ijkm\bullet}) \qquad (3.3)$$

  Dimana, $n_{ijk+1(diproduksi)} \geq n_{ij(k+1)} * (1 + dr_{ijk+1m\bullet})$, sehingga persamaan (3.3) menyatakan bahwa jumlah komponen ke-*j* dari item *i* pada operasi ke-*k* yang harus diproduksi, pada kondisi suatu item harus diproses lebih dari satu kali pada mesin yang berbeda, ditentukan oleh jumlah komponen pada operasi ke-*k*+1 dan *defect rate* (*dr*) untuk

semua mesin *m* yang identik yang tersedia dalam tahapan produksi pada operasi ke-*k+1*, dengan pembulatan dilakukan ke atas.

- Fungsi kendala hubungan jumlah komponen atau *part* dalam *batch* dengan jumlah komponen akhir.

$$\sum_{u=1}^{N} Q_{ij}[u] = n_{ij} \; ; \forall \, i, j \tag{3.4}$$

Persamaan (3.4) menyatakan keseimbangan material, artinya bahwa jumlah komponen atau item yang diproduksi pada seluruh *batch* sebanyak *N batch* sama dengan jumlah komponen atau item bersangkutan, sebelum dilakukan pemecahan kedalam *batch-batch* produksi.

- Fungsi kendala kelayakan waktu memulai penjadwalan

$$B_{ij1\bullet\bullet\bullet}[N] \geq 0 \; ; \forall \, i, j \tag{3.5}$$

Persamaan (3.5) menjamin bahwa waktu dimulai pemprosesan *batch* pertama dari komponen p_{ij} harus lebih besar atau sama dengan nol, yang berarti pekerjaan tersebut layak dikerjakan didalam horizon waktu perencanaan.

- Fungsi kendala hubungan waktu penyelesaian komponen terhadap induk langsungnya.

$$C_{ijhl+1\bullet\bullet} - B_{ij1l\bullet\bullet} \leq 0 \; ; \forall \, i, j, k, l \tag{3.6}$$

Persamaa (3.8) menjamin bahwa pemproses *batch* pertama operasi pertama *k=1*untuk komponen p_{ij} pada level ke-*l* sebagai induk langsung dari komponen p_{ij} pada level ke-*l+1*, dapat dimulai apabila operasi terakhir *k=h* komponen p_{ij} telah selesai dikerjakan, disemua mesin.

- Fungsi kendala hubungan antar pemprosesan operasi *batch*.

$$C_{ij(k-1)l\bullet\bullet\bullet}[u] - B_{ijkl\bullet\bullet}[u] \leq 0 \; ; \forall \, i, j, k, l, u \tag{3.7}$$

Persamaan (3.9) menyatakan pemproses *batch* ke-*u* suatu operasi bisa dilakukan bila pemproses pada operasi sebelumnya telah selesai dikerjakan.

- Fungsi kendala hubungan pemrosesan antar *batch*.

$$B_{ijklm\bullet}[u] \le B_{ijklm\bullet}[u-1] - \left(Q_{ijkl}[u] * t_{ijkm\bullet}\right) - s_{ijkm\bullet}\ ;$$
$$\forall\ i, j, k, l, m, u \qquad (3.8)$$

Persamaan (3.10) menyatakan pemprosesan suatu *batch* ke-*u* harus segera dilakukan bila *batch* ke *u-1* atau batch sebelumnya telah selesai diproses.

- Fungsi kendala ketepatan pemenuhan *due date.*

$$B_{i0he\bullet\bullet}[1] = d_{i0} - \left(Q_{i0h0[1]} * t_{i0h\bullet\bullet}\right);\ \forall\ i \qquad (3.9)$$

Persamaan (3.11) menyatakan bahwa *batch* pertama operasi terakhir *k=h* dari tiap item *i*, pada level terakhir *l=0* harus selesai tepat pada saat *due date*-nya.

- Fungsi kendala untuk menjamin bahwa suatu mesin hanya dapat memproses satu *batch* operasi saja pada waktu tertentu.

$$C_{\bullet\bullet\bullet\bullet mn[w]} - C_{\bullet\bullet\bullet\bullet mn[y]} + \alpha\left(X_{wy}\right) \ge \left(Q_{\bullet\bullet\bullet\bullet[w]} * t_{\bullet\bullet\bullet\bullet mn}\right);$$
$$\forall\ m, n \qquad (3.10)$$

$$C_{\bullet\bullet\bullet\bullet mn[y]} - C_{\bullet\bullet\bullet\bullet mn[w]} + \alpha\left(1 - X_{wy}\right) \ge \left(Q_{\bullet\bullet\bullet\bullet[y]} * t_{\bullet\bullet\bullet\bullet mn}\right);$$
$$\forall\ m, n \qquad (3.11)$$

Persamaan (3.12 dan 3.13) menyatakan waktu penyelesaian *batch* pada posisi *w*, apabila dikurangi dengan waktu penyelesaian *batch* pada posisi berikutnya (posisi *y*) di tambah dengan bilangan positif yang besar (α) yang lebih besar atau sama dengan waktu diperlukan untuk menyelesaikan jumlah part dalam *batch* pada posisi ke-*w*. kondisi ini berlaku untuk *batch* dengan urutan lebih awal diproses lebih dahulu pada mesin-*m* ke-*n*. apabila urutan pemprosesan dibalik akan berlaku persamaan (3.13).

- Fungsi kendala untuk kondisi variabel.

$$X_{wy} \in \{0,1\},\ Q_{ijkl[u]} > 0\text{, dan } N_{ijk} \ge 1, \text{integer} \qquad (3.12)$$

Persamaan (3.14) menyatakan nilai X_{wy} adalah 1 jika *batch* operasi ke-*k* diposisi *w* yaitu $L_{[w]}$ mendahului $L_{[y]}$, atau bernilai 0 jika sebaliknya, ukuran *batch* harus bernilai positif, dan jumlah *batch* merupakan bilangan *integer* yang bernilai lebih besar atau sama dengan satu.

Formulasi dari permasalahan tersebut merupakan model yang kompleks karena banyak variabel terkait. Oleh karena itu, diusulkan penggunakan pendekatan heuristik untuk mendapatkan solusi, dengan mengubah variabel-menjadi parameter. Dimana variabel-variabel yang dapat diubah menjadi parameter adalah jumlah *batch* item dan komponen-komponennya serta urutan pengerjaan *batch-batch* tersebut. Dengan demikian, sebelum menyelesaikan formulasi matematis di atas, diperlukan sejumlah iterasi yang jumlahnya tergantung jumlah, jenis, dan *level* pada struktur produk item serta jumlah stasiun kerja yang dilalui (*multi-stage*).

Untuk mengubah beberapa variabel (jumlah *batch* item dan komponen-komponennya serta urutan pengerjaan *batch-batch* bersangkutan) menjadi parameter dan untuk menjawab kedatangan pesanan baru, disusun serangkaian algoritma yang dapat mengontrol jalannya proses pencarian solusi sehingga lebih terarah.

Secara garis besar, algoritma yang dikembangkan bekerja dengan tahapan sebagai berikut:

- Menentukan jumlah dan ukuran *batch* serta menjadwalkannya untuk pesanan-pesanan yang telah ada (saat $t = 0$) dengan cara:
  - Mengurutkan semua item sesuai dengan urutan *due-date* secara menurun atau tidak menaik, sebagaimana hasil penelitian Halim 1993).
  - Penentuan jumlah dan ukuran *batch* untuk setiap item, dimulai dari *level 0* sampai pada *level* terendah dengan mengakomodasi tingkat cacat dari setiap proses. Penentuan ini dimulai dengan $N_{ijk} = 2$ sampai didapatkan *total actual flow time* yang minimum.
  - Penjadwalan secara *backward* dimulai dari *due-date* terbesar.
  - Pengurutan pengerjaan operasi semua komponen penyusun item pada setiap mesin dan pemecahan *batch* pada masing-masing operasi tersebut dimulai dari *batch* yang paling dekat dengan *due-date* yang memberikan hasil yang lebih baik seperti diperlihatkan pada penelitian-penelitian sebelumnya, sesuai dengan Teorema 1 dan Teorema 2.

- Penggabungan jadwal pada semua mesin dengan menggunakan teknik penyisipan.

❖ Gambarkan jadwal akhir untuk tiap item dan komponen dengan memperhatikan ketersediaan mesin.

❖ Apabila terjadi kedatangan pesanan baru, untuk menyatakan pesanan baru tersebut diterima atau ditolak dilakukan penjadwalan ulang atau *rescheduling* dengan menggunakan teknik penyisipan.

Algoritma yang diusulkan untuk menyelesaikan masalah penjadwalan *batch* tersebut bekerja dalam dua tahapan algoritma, yaitu algoritma untuk menentukan jumlah dan ukuran *batch* serta menjadwalkannya, yang menggunakan aturan penentuan da pemilihan ukuran *batch* dari Halim et al,(1994) dan teknik penyisipan untuk penggabungan jadwal pada setiap mesin untuk membentuk jadwal akhir.

3.3.1 Algoritma Penyelesaian Pemenuhan Due Date Kondisi Statis

Didalam proses penyelesaian permasalahan, dilakukan beberapa tahapan pemprosesan pencarian solusi sampai terbentuknya jadwal akhir. Adapun *Flow chart* algoritma yang dikembangkan untuk penyelesaikan permasalah pemenuhan *due date* pada kondisi statis, seperti ditunjukan pada gambar 3.3, berikut ini.

Algoritma Penjadwalan *Batch* pada *Job Shop* Statis.Pesanan-pesanan yang telah ada pada $t = 0$ dijadwalkan mengikuti aturan, sebagai berikut:

Langkah 0. Tentukan $P(p_{i0})$ yaitu set dari item-item yang akan diproduksi.

Langkah 1. Urutkan item-item pada $P(p_{i0})$ sesuai dengan urutan *due date* secara menurun atau tidak menaik dan nyatakan sebagai $P'(p_{i0})$.

Langkah 2. Tetapkan item yang menempati urutan pertama pada $P'(p_{i0})$ sebagai item $\alpha = 1$ (dengan indeks $\alpha = 1, 2, ..., \beta$).

Langkah 3.Buat struktur dari item bersangkutan dan gambarkan *digraph* dari produk akhir dan semua komponennya. Tiap tahapan pada *digraph* dinyatakan dengan *level*, dimana penomoran *level* produk dimulai dengan menyatakan *level 0* untuk produk akhir, kemudian nomor *level* bertambah satu pada setiap langkah mundur hingga mencapai komponen pada *level* terendah (L).

*Langkah 4.*Tentukan jenis komponen yang akan diproduksi dan jumlah mesin yang digunakan $(p_{ij}$ dan $m)$.

Langkah 5. Untuk produk akhir, tentukan jumlah produk akhir yang harus dibuat dengan mengakomodasi tingkat cacat dari tiap proses, semua operasi untuk produk akhir, waktu *set-up* dan waktu proses operasi tersebut di setiap mesin, dan *due-date* produk akhir $(n_{i0}, o_{i0k0m}, s_{i0k0m}, t_{i0k0m},$ dan $d_{i0})$. Jika produk akhir diproses pada dua mesin atau lebih, maka kuantitas pada posisi *k* ditentukan dari kuantitas pada posisi *k+1*.

*Langkah 6.*Untuk setiap komponen penyusun produk akhir, tentukan induk komponen bersangkutan hingga produk akhir dan banyaknya komponen bersangkutan yang dibutuhkan untuk membuat satu unit induk langsung $(Z(p_{ij})$dan $H_{ij})$.

Langkah 7. Untuk tiap komponen, tentukan jumlah komponen yang harus diproduksi dengan mengakomodasi tingkat cacat tiap proses, operasi-operasi untuk komponen bersangkutan, dan waktu *set-up* dan waktu proses semua operasi tersebut di setiap mesin $(n_{ij}, o_{ijklml}, s_{ijkm\bullet},$ dan $t_{ijkm\bullet})$ dengan $n_{ij} = n_{i0} * H_{ij}\left(\prod_{p_{ik} \in Z(p_{ij})} H_{ik}\right)$

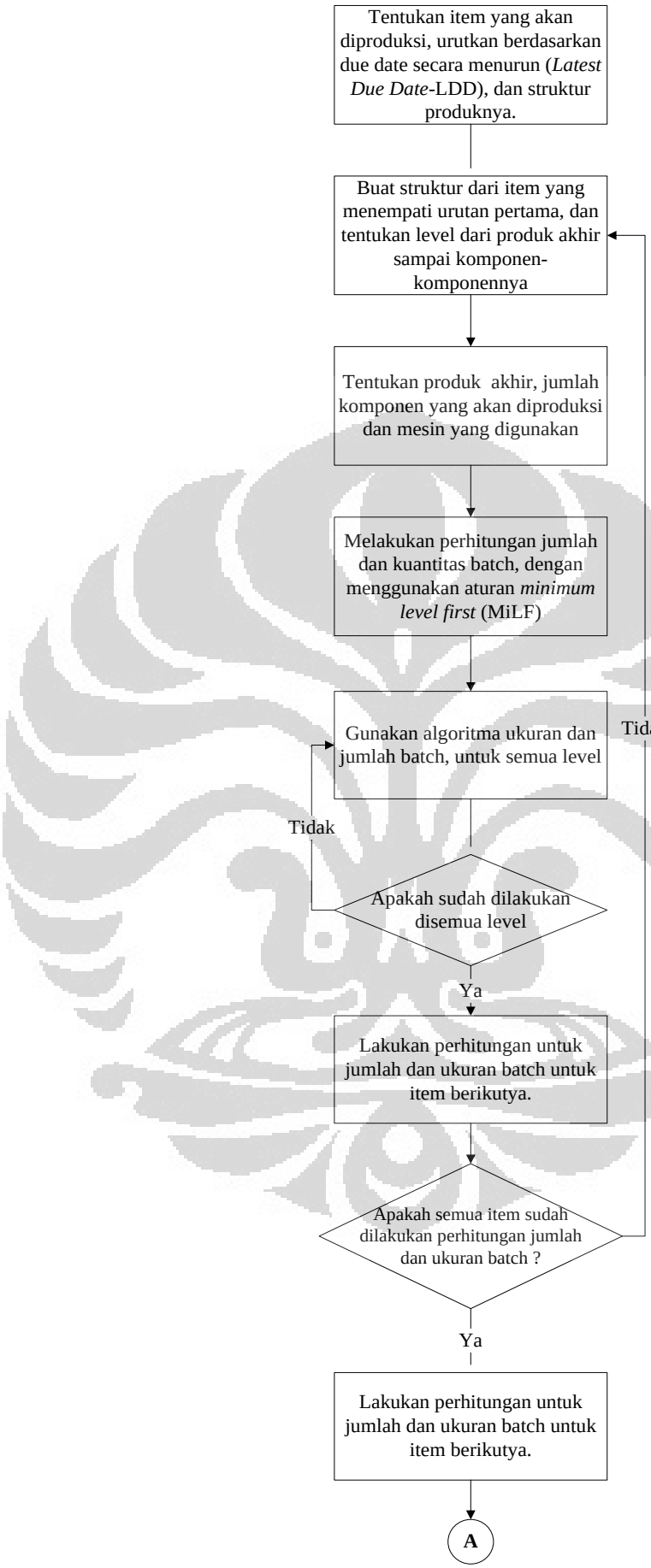
dimana $\underset{p_{ik} \in Z(p_{ij})}{H_{ik}} \geq 1$. dimana kuantitas pada urutan *k* ditentukan pada kebutuhan diurutan *k+1*.

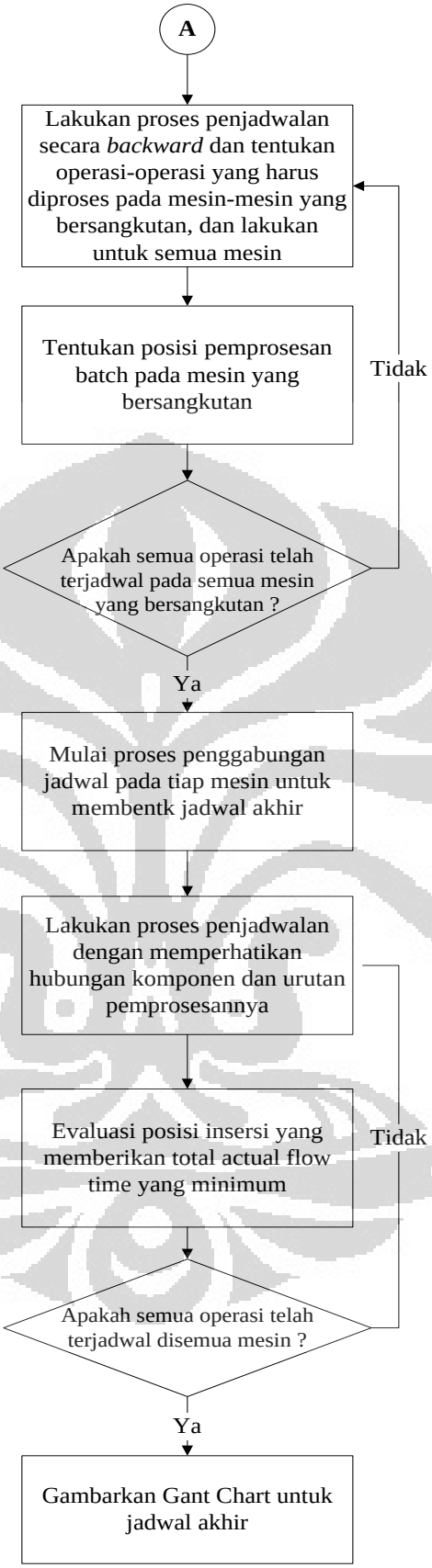
Langkah 8. Karena pendekatan yang digunakan adalah pendekatan mundur (*backward approach*), maka tahapan pemrosesan mengikuti aturan *minimum level first* (MiLF), dimulai dari *level 0* atau produk akhir.

Langkah 9. Gunakan **[Sub-Algoritma Penentuan Jumlah dan Ukuran *Batch*]**. Lanjutkan ke *Langkah 10.*

Langkah 10. Set $l = l + 1$.

- Jika $l \leq L$, kembali ke *Langkah 9.*
- Jika $l > L$, lanjutkan ke *Langkah 11*.





Gambar 3.3 Flow Chart Pemenuhan Due Date Kondisi Statis

- Jika tidak, lakukan pemecahan *batch* untuk komponen tersebut dengan menggunakan hasil yang didapatkan sebelumnya dari **[Sub-Algoritma Penentuan Jumlah dan Ukuran *Batch*]** dan jadwalkan *batch-batch* komponen tersebut dengan urutan berdasarkan Teorema 1. Lanjutkan ke *Langkah 28*.

Langkah 22. Kelompokkan komponen yang sejenis dalam satu *batch*, $N_{ijk} = 1$ sesuai dengan mesinnya dan dengan Teorema 2 tentukan urutan yang menghasilkan total waktu tinggal aktual yang paling minimum. Beri indeks $x = 1,2,...,y$.

Langkah 23. Pilih *batch* untuk komponen yang berada paling dekat dengan *due-date*, nyatakan sebagai *batch* komponen $x = 1$.

Langkah 24. Lakukan pemecahan *batch* untuk komponen x tersebut dengan menggunakan hasil yang didapatkan sebelumnya dari **[Sub-Algoritma Penentuan Jumlah dan Ukuran *Batch*]**.

Langkah 25. Jadwalkan *batch-batch* komponen tersebut dengan urutan berdasarkan Teorema 1.

Langkah 26. Set $x = x + 1$.

- Jika $x \le y$, kembali ke *Langkah 24*.
- Jika $x > y$, lanjutkan ke *Langkah 27*.

Langkah 27. Set $\alpha = \alpha + 1$.

- Jika $\alpha \le \beta$, kembali ke *Langkah 21*.
- Jika $\alpha > \beta$, lanjutkan ke *Langkah 28*.

Langkah 28. Ulangi *Langkah 16* sampai seluruh operasi pada mesin bersangkutan telah dijadwalkan. Jika semua operasi pada mesin bersangkutan telah terjadwal ulangi *Langkah 14*.

Langkah 29. Penggabungan jadwal pada tiap-tiap mesin untuk membentuk S_{akhir}. Untuk tiap-tiap mesin, lakukan *Langkah 30 - 38*.

Langkah 30.Tempatkan operasi terakhir untuk produk akhir sedemikian sehingga saat selesai operasi bersangkutan tepat pada *due-date* produk akhir.

Langkah 31.Tentukan $O'(o_{ijklm})$ yaitu set operasi-operasi dari komponen komponen sesuai dengan urutan pemprosesan pada mesin

bersangkutan (kecuali operasi yang telah ditempatkan pada *Langkah 30*). Beri indeks $x = 1,2,...,y$.

Langkah 32. Mulai dengan operasi yang menempati urutan pertama pada $O'(o_{ijklm})$ dan nyatakan sebagai operasi $x = 1$. Penempatan pekerjaan dilakukan dari mesin *m* ke *1* dan seterusnya

Langkah 33. Apakah pada saat t tertentu, terdapat operasi yang telah terjadwal ?

- Jika ya, periksa apakah ada mesin ke n yang masih tersedia.apabila tersedia jadwalkan batch-batch operasi *x*. Jika tidak tersedia lanjutkan ke *Langkah 34*.
- Jika tidak, jadwalkan *batch-batch* operasi x. Lanjutkan ke *Langkah 38*.

Langkah 34. Apakah indeks produk akhir dari komponen untuk operasi x sama dengan indeks untuk operasi terjadwal, pada mesin *m* ke *n* ?

- Jika ya, lanjutkan ke *Langkah 35*.
- Jika tidak, lanjutkan ke *Langkah 37*.

Langkah 35. Apakah indeks komponen untuk operasi x sama dengan indeks untuk operasi terjadwal, pada mesin *m* ke *n* ?

- Jika ya, jadwalkan *batch-batch* operasi x di belakang (mengikuti) *batch-batch* operasi yang telah terjadwal tersebut. Lanjutkan ke *Langkah 38*.
- Jika tidak, lanjutkan ke *Langkah 36*.

Langkah 36. Apakah komponen untuk operasi x berada pada *level* yang sama dengan komponen untuk operasi terjadwal ?

- Jika ya, lanjutkan ke *Langkah 37*.
- Jika tidak, jadwalkan *batch-batch* operasi x di belakang (mengikuti) *batch-batch* operasi yang telah terjadwal tersebut. Lanjutkan ke *Langkah 38*.

Langkah 37. Evaluasi posisi penyisipan yang mungkin untuk *batch-batch* operasi x yaitu di depan (mendahului) atau di belakang (mengikuti) *batch-batch* operasi yang telah terjadwal tersebut untuk semua mesin *m*. Pilih posisi penyisipan yang menghasilkan total waktu tinggal aktual paling minimum, lalu jadwalkan *batch-batch* operasi x berikutnya. Jika terdapat operasi yang terjadwal pada *t* tertentu untuk semua

mesin *m* ke *n*. Kembali ke *Langkah 34*. Jika semua batch operasi *x* telah terjadwal lanjut ke Langkah 38,

Langkah 38. Set $x = x + 1$.

- Jika $x \leq y$, kembali ke *Langkah 33*.
- Jika $x > y$, lanjutkan ke *Langkah 39*.

Langkah 39.Gambarkan *Gantt-Chart* untuk jadwal yang terbentuk (S_{akhir}).

3.3.2 Algoritma Penentuan Ukuran dan Jumlah *Batch*

Secara garis besar algoritma penentuan ukuran dan jumlah *batch* bekerja dengan diagram alir, seperti ditunjukan pada gambar 3.4.

Langkah 1. Tentukan $L(p_{ij})$ yaitu set dari komponen-komponen item i yang berada pada *level* l struktur produk item i tersebut. Beri indeks $x = 1,2,...,y$.

Langkah 2. Untuk masing-masing komponen tersebut, tentukan pula $M(p_{ij})$ yaitu set dari mesin-mesin yang digunakan untuk pemrosesan komponen p_{ij} bersangkutan.

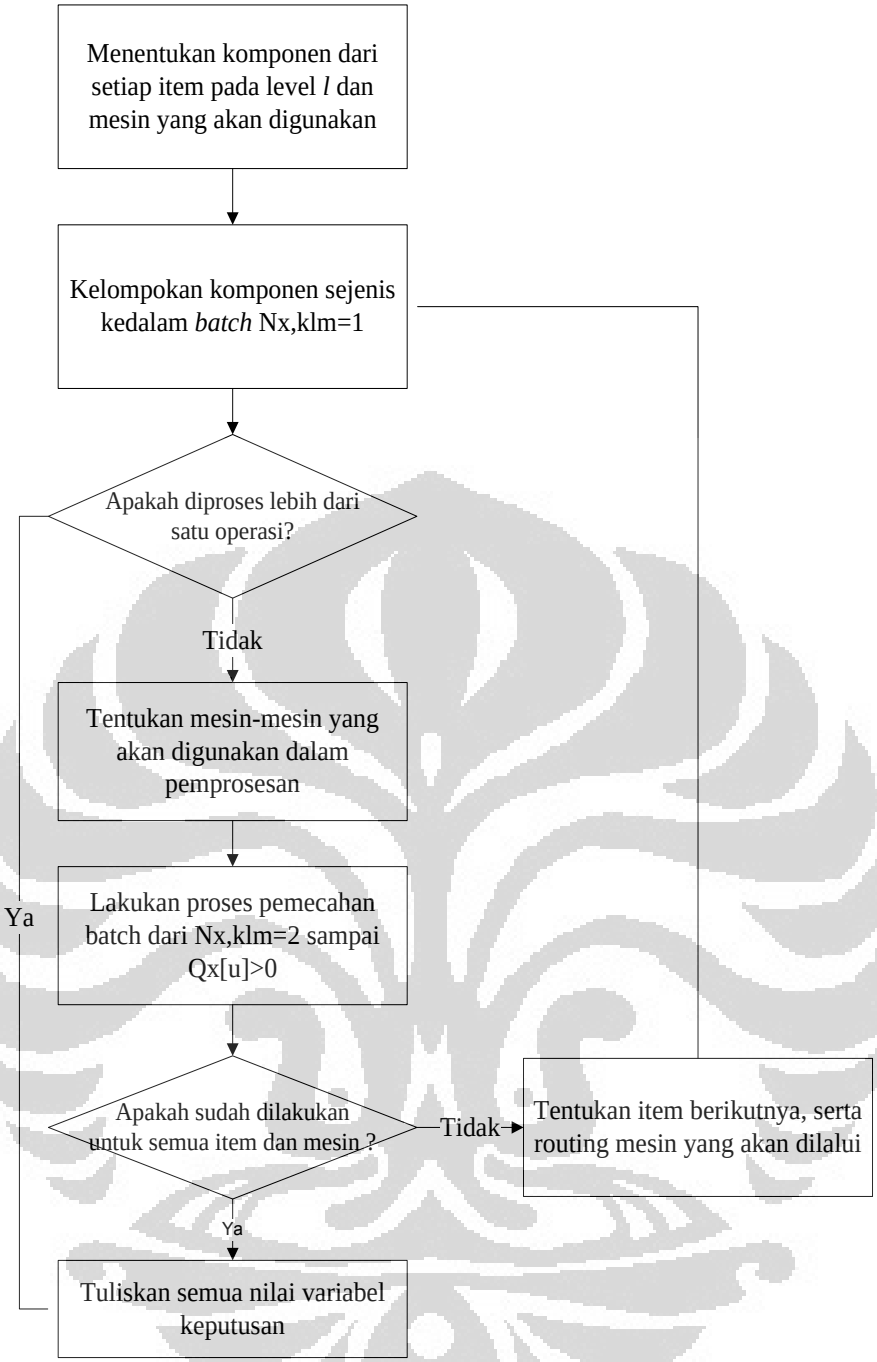
Langkah 3.Kelompokan komponen yang sejenis dalam satu *batch*, $N_{x,klm} = 1$, untuk $x = 1,2,...,y$.

Langkah 4. Pilih *batch* untuk komponen pertama pada $L(p_{ij})$, nyatakan sebagai *batch* komponen $x = 1$.

Langkah 5. Jadwalkan operasi-operasi pada komponen x tersebut pada tiap mesin anggota $M(p_{ij})$ sesuai dengan *routing* operasinya dan hitung flow timenya. Apakah hanya diproses pada satu mesin ?

- Jika Ya, maka $N_{x,klm} = 1$ optimal, lanjut ke langkah 14
- Jika tidak, lanjut ke langkah 6

Langkah 6. Set o= mesin yang menjadi anggota pertama pada $M(p_{ij})$, gunakan waktu *set-up* dan waktu proses operasi komponen pada mesin tersebut sebagai parameter.



Gambar 3.4 Diagram Alir Algoritma Penentuan Jumlah dan Ukuran *Batch*

Langkah 7.Mulai memecah *batch* untuk komponen x secara bertahap untuk setiap mesin yan digunakan, set $N_{x,klm} = 2$.

Langkah 8. Untuk *batch* $u = 1$ hingga $N_{x,klm}$, hitung ukuran *batch*, $Q_{x[u],klm}$ dengan rumus sebagai berikut.

$$Q_{x[u,],klm} = \left(n_{x,km} / N_{x,klm}\right) + 0.5 * \left(N_{x,klm} + 1\right)\left(su_{xo,km} / t_{xo,km}\right) - \left(su_{xo,km} / t_{xo,km}\right) * u$$

Langkah 9. Apakah $Q_{x[u]} > 0$?

- Jika ya, lanjut ke langkah 10.
- Jika tidak, set $N_{x(opt)} = N_x - 1$ dan lanjutkan ke *Langkah 13.*

Langkah 10. Apakah operasi tersebut berada pada operasi terakhir ?

- Ya, lanjut ke langkah 12.
- Tidak, lanjut ke langkah 11.

Langkah 11. Apakah $F^a_{N_{x,klm}} \leq F^a_{N_{x,klm}-1}$?

- Jika ya, lanjutkan ke *Langkah 11*.
- Jika tidak, set $N_{x,klm(opt)} = N_{x,klm} - 1$, lanjutkan ke *Langkah 13*.

Langkah 12. Set $N_{x,klm} = N_{x,klm} + 1$, dan kembali ke *Langkah 8*.

Langkah13. Set o = mesin yang menjadi anggota berikutnya pada $M(p_{ij})$, gunakan waktu *set-up* dan waktu proses operasi komponen pada mesin tersebut sebagai parameter.

- Jika $o \leq m$, kembali ke *Langkah 7*.
- Jika $o > m$, berhenti. Pilih solusi optimal dari beberapa solusi alternatif dan tuliskan semua nilai variabel keputusan $\left(N_{x,klm} \text{ dan } Q_{x,klm}[u]\right)$ pada masing-masing mesin. Kemudian lanjutkan ke *Langkah 14*.

Langkah 14. Set $x = x + 1$.

- Jika $x \leq y$, kembali ke *Langkah 5*.
- Jika $x > y$, berhenti. Tuliskan semua nilai variabel keputusan untuk tiap komponen.

3.3.3 Algoritma Pemenuhan *Due Date* Kondisi Dinamis

Jika terjadi kedatangan pesanan-pesanan baru pada saat produksi sedang berjalan, maka digunakan algoritma untuk menyelesaikan masalah penjadwalan *job shop* kondisi dinamis. Dimana setiap pesanan baru tersebut memiliki *due-date* yang berbeda atau *multi-due date* dengan *due date* lebih besar dari pada *due date* item-item yang telah terjadwal sebelumnya. Algoritma merupakan sekumpulan langkah-langkah yang dilakukan dengan adanya pesanan-pesanan baru pada saat T=*A*. Dengan prosedur penyelesaian secara garis besar ditunjukan seperti pada gambar 3.5, berikut ini langkah-langkah penyelesaian kondisi dinamis.

Langkah 0. Tentukan jumlah pesanan baru dan *due-date* masing-masing pesanan tersebut. Nyatakan $I(p_{i0})$ yaitu set dari pesanan-pesanan baru dengan urutan *due-date* yang tidak menurun dan beri indeks $\gamma = 1,2,...,\sigma$.

Langkah 1. Mulai dengan pesanan baru yang menempati urutan pertama pada $I(p_{i0})$ dan nyatakan sebagai pesanan $\gamma = 1$.

Langkah 2. Susun jadwal terpisah untuk pesanan baru dengan menggunakan **[Algoritma Penemenuhan *Due Date* pada *Job Shop* Statis]** untuk kasus item tunggal.

Langkah 3. Pada jadwal yang telah ada akan dilakukan penyisipan operasi operasi pesanan baru pada posisi yang memungkinkan secara *backward*.

Langkah 4. Untuk tiap mesin, lakukan *Langkah 5 - 13*. Dimulai dari mesin yang memproses operasi paling terakhir terlebih dahulu pada dengan aturan *Minimum Level First* (*MiLF*).

Langkah 5.Untuk masing-masing mesin tersebut, tentukan $I(o_{ijklm})$ yaitu set operasi-operasi dari komponen-komponen pesanan baru yang akan dipenyisipankan pada mesin bersangkutan, serta akomodasi ketersediaan mesin tersebut, sesuai dengan urutan pada jadwal hasil *Langkah* 2. Beri indeks $x = 1,2,...,y$.

Langkah 6. Periksa apakah ada *batch* yang belum selesai diproses, untuk semua mesin identik dan tentukan .

- Jika ya, lanjutkan ke *Langkah 7*.
- Jika tidak, lanjutkan ke *Langkah 9*.

Langkah 7. Periksa apakah *batch* yang belum selesai diproses tersebut adalah *batch* nomor 1.

- Jika ya, maka selesaikan pemprosesan *batch* tersebut. Lanjutkan ke *Langkah 8*.
- Jika tidak, selesaikan pemrosesan sampai *batch* nomor 1. Kemudian lanjutkan ke *Langkah 8*.

Langkah 8. Simpan saat selesai pemprosesan *batch* dan nyatakan sebagai $(T_{av})_{mn}$, yaitu titik waktu dimana mesin m ke *n* tersedia untuk melakukan pemprosesan operasi-operasi berikutnya.

Langkah 9. Mulai dengan operasi yang menempati urutan pertama pada $I(o_{ijklm})$ dan nyatakan sebagai operasi $x = 1$.

Langkah 10. Apakah pada t tertentu, terdapat operasi yang telah terjadwal untuk semua mesin?

- Jika ya, lanjutkan ke *Langkah 11*.
- Jika tidak, jadwalkan operasi x. Simpan saat mulai operasi tersebut dan nyatakan sebagai t_x. Lanjutkan ke *Langkah 13*.

Langkah 11. Evaluasi posisi penyisipan yang mungkin untuk operasi x yaitu di depan (mendahului) atau di belakang (mengikuti) operasi yang telah terjadwal tersebut. Pilih posisi penyisipan yang menghasilkan total waktu tinggal aktual yang paling minimum. Catatan: penyisipan tidak diantara *batch-batch* operasi dari komponen yang sama.

Langkah 12. Simpan saat mulai operasi x dan nyatakan sebagai t_x.

Langkah 13. Set $x = x + 1$.

Catatan: posisi penyisipan untuk operasi-operasi ini harus memiliki saat mulai yang lebih kecil dari operasi pada urutan sebelumnya atau $(t_{x+1} < t_x)$.

- Jika $x \leq y$, kembali ke *Langkah 10*.
- Jika $x > y$, lanjutkan ke *Langkah 14*.

Langkah 14. Lakukan evaluasi terhadap $(T_{av})_{mn}$ untuk tiap-tiap mesin, apabila terlanggar maka pesanan baru ditolak.

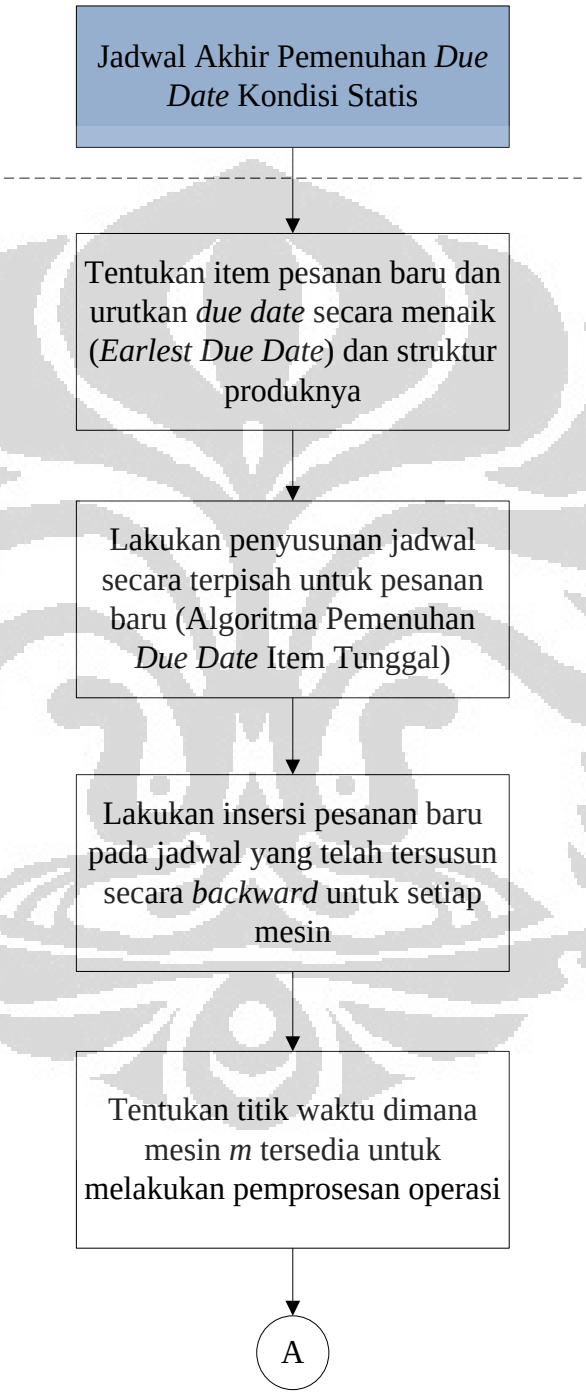
Langkah 15. Apabila $(T_{av})_{mn}$ tidak terlanggar, maka gabungkan jadwal pada tiap-tiap mesin untuk membentuk S_{akhir}.

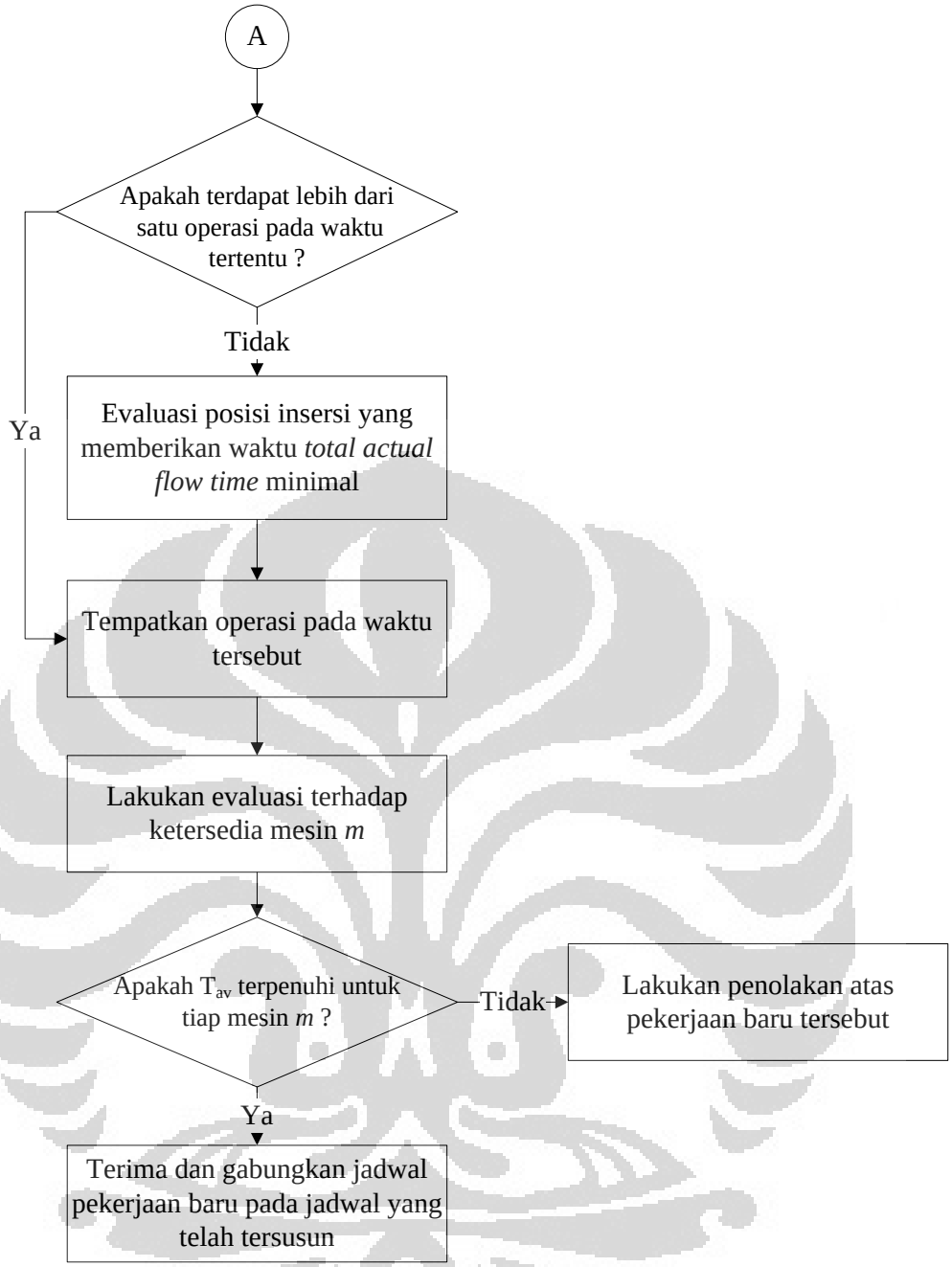
Langkah 16. Set $\gamma = \gamma + 1$.

- Jika $\gamma \leq \sigma$, kembali ke *Langkah 2*.

- Jika $\gamma > \sigma$, lanjutkan ke *Langkah 17.*

Langkah 17. Kelompokkan pesanan-pesanan yang diterima pada O_1 dan tampilkan S_{akhir}. Sementara, pesanan-pesanan yang ditolak dikelompokan pada O_2.





Gambar 3.5 Diagram Alir Pemenuhan *Due Date* Kondisi Dinamis

3.3.4 Algoritma Pemenuhan *Due Date Job Shop* Statis Item Tunggal

Secara garis besar algoritma pemenuhan *due date job shop* pada kasus item tunggul kondisi statis bekerja dengan diagram alir, seperti ditunjukan pada gambar 3.6.Pesanan yang telah ada pada $t = 0$ dijadwalkan mengikuti aturan, sebagai berikut.

Langkah 0. Buat struktur dari item bersangkutan dan gambarkan *digraph* dari produk akhir dan komponen-komponennya. Tiap tahapan pada

digraph dinyatakan dengan *level*, dimana penomoran *level* produk dimulai dengan menyatakan *level 0* untuk produk akhir, kemudian nomor *level* bertambah satu pada setiap langkah mundur hingga mencapai komponen pada *level* terendah (L).

Langkah 1. Tentukan jenis komponen yang akan diproduksi dan jumlah mesin yang digunakan $(p_{ij}$ dan $m)$.

Langkah 2. Untuk produk akhir, tentukan jumlah produk akhir yang harus dibuat dengan mengakomodasi tingkat cacat dari tiap proses, operasi-operasi untuk produk akhir, waktu *set-up* dan waktu proses operasi-operasi tersebut di masing-masing mesin, dan *due-date* produk akhir $(n_{i0}, o_{i0klm}, s_{i0klm}, t_{i0klm},$ dan $d_{i0})$.

Langkah 3. Untuk komponen-komponen produk akhir, tentukan induk induk komponen bersangkutan hingga produk akhir dan banyaknya komponen bersangkutan yang dibutuhkan untuk membuat satu unit induk langsung $(Z(p_{ij})$ dan $H_{ij})$.

Langkah 4. Untuk tiap komponen, tentukan jumlah komponen yang harus diproduksi dengan mengakomodasi tingkat cacat tiap proses, operasi-operasi untuk komponen bersangkutan, dan waktu *set-up* dan waktu proses operasi-operasi tersebut di masing-masing mesin $(n_{ij}, o_{ijklm}, s_{ijklm},$ dan $t_{ijklm})$ dengan $n_{ij} = n_{i0} * H_{ij}\left(\prod_{p_{iq} \in Z(p_{ij})} H_{iq}\right)$ dimana $\prod_{p_{iq} \in Z(p_{ij})} H_{iq} \geq 1$. Dimana kebutuhan n_{ij} yang harus diproduksi pada urutan *k* ditentukan berdasarkan kebutuhan pada *k+1*.

Langkah 5. Karena pendekatan yang digunakan adalah pendekatan mundur (*backward approach*), maka tahapan pemprosesan mengikuti aturan *minimum level first* (MiLF), dimulai dari *level 0* atau produk akhir.

Langkah 6. Gunakan **[Sub-Algoritma Penentuan Jumlah dan Ukuran *Batch*]**. Lanjutkan ke *Langkah 7*.

Langkah 7. Tentukan $M(p_{il})$ yaitu set dari mesin-mesin yang digunakan untuk pemrosesan komponen-komponen yang berada pada *level l* struktur produk akhir i.

Langkah 8. Untuk masing-masing mesin tersebut, tentukan pula $O(o_{ijklm})$ yaitu set operasi-operasi dari komponen-komponen yang akan diproses pada mesin bersangkutan.

Langkah 9. Untuk tiap mesin dimulai mesin dengan indeks terkecil (*m=1)*, lakukan *Langkah 10 - 17*.

Langkah 10. Cari operasi-operasi paling hilir (operasi paling akhir) yang belum dijadwalkan.

Langkah 11. Apakah pada saat t tertentu, terdapat lebih dari satu operasi yang membutuhkan mesin bersangkutan ?

- Jika ya, lanjutkan ke *Langkah 12*.
- Jika tidak, lakukan pemecahan *batch* untuk komponen tersebut dengan menggunakan hasil yang didapatkan sebelumnya dari **[Sub-Algoritma Penentuan Jumlah dan Ukuran *Batch*]** dan jadwalkan *batch-batch* komponen tersebut dengan urutan berdasarkan Teorema 1. Jika semua operasi telah terjadwal, Lanjutkan ke *Langkah 18*.

Langkah 12. Kelompokkan komponen yang sejenis dalam satu *batch*, $N_{ij} = 1$, sesuai dengan mesinnya dan dengan Teorema 2 tentukan urutan yang menghasilkan total waktu tinggal aktual yang paling minimum. Beri indeks $x = 1,2,...,y$.

Langkah 13. Pilih *batch* untuk komponen yang berada paling dekat dengan *due-date*, nyatakan sebagai *batch* komponen $x = 1$.

Langkah 14. Lakukan pemecahan *batch* untuk komponen x tersebut dengan menggunakan hasil yang didapatkan sebelumnya dari **[Sub-Algoritma Penentuan Jumlah dan Ukuran *Batch*]**.

Langkah 15. Jadwalkan *batch-batch* komponen tersebut dengan urutan berdasarkan Teorema 1.

Langkah 16. Set $x = x + 1$.

- Jika $x \leq y$, kembali ke *Langkah 14*.

- Jika $x > y$, lanjutkan ke *Langkah 17*.

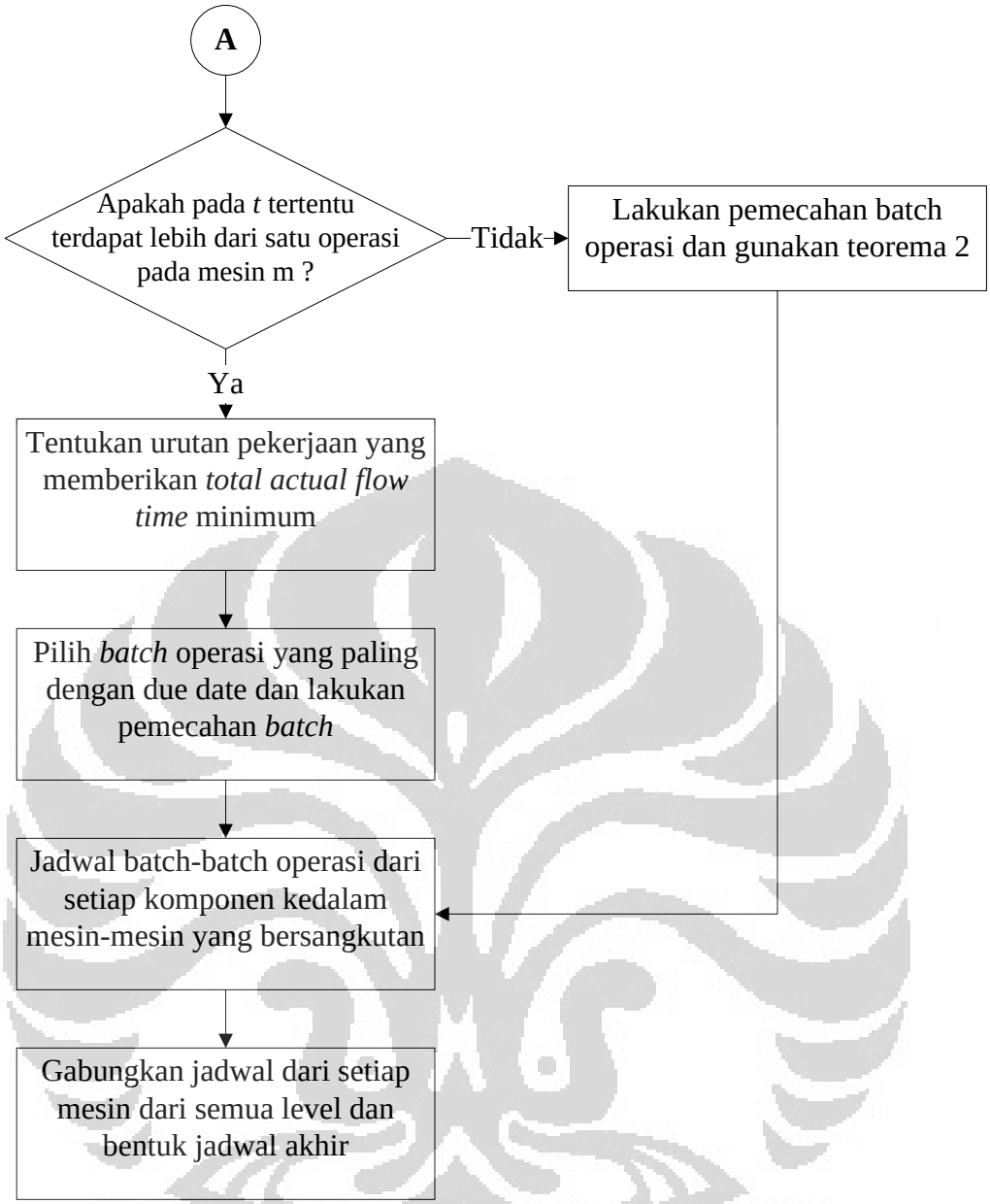
Langkah 17. Ulangi *Langkah 9* sampai seluruh operasi pada mesin bersangkutan telah dijadwalkan.

Langkah 18. Gabungkan jadwal pada tiap-tiap mesin untuk membentuk S_l yaitu jadwal parsial untuk *level* l struktur produk akhir $_i$.

Langkah 19. Set $l = l + 1$.

- Jika $l \leq L$, kembali ke *Langkah 6*.
- Jika $l > L$, lanjutkan ke *Langkah 20*.

Langkah 20. Gabungkan jadwal pada tiap-tiap *level* untuk membentuk S_{akhir} dimana $S_{akhir} = \{S_0, S_1, ..., S_l\}$ secara *backward*.



Gambar 3.6 Diagram Alir Pemenuhan *Due Date* Job Shop Item Tunggal

3.4 Model Penjadwalan Penentuan *Due Date*

Kondisi permasalahan pada model ini, pada dasarnya berlawanan dengan kondisi permasalahan pemenuhan *due date* (model satu), di mana dalam model ini, *due date* pada masing-masing item tidak diketahui. Karena *due date* diasumsikan dapat dinegosiasikan dengan konsumen. Sehingga fungsi kendala pada persamaan (3.11) pada model ini, dapat diabaikan. Fungsi tujuan untuk kondisi penentuan diperoleh dengan memodifikasi persamaan (3.1), sehingga fungsi tujuan untuk penentuan *due date* sebagai berikut:

$$\text{Minimasi } TF = \sum_{i=1}^{r}\left(C_{i0\bullet 0\bullet\bullet} - B_{i\bullet 1e\bullet\bullet[N]}\right) \tag{3.13}$$

Persamaan (3.13) menyatakan minimasi total waktu tinggal aktual untuk semua item, dengan $B_{i\bullet 1e\bullet\bullet[N]}$ menyatakan saat mulai *batch* pertama untuk operasi pertama dari komponen yang berada pada *level e* struktur produk item *i* yang terjadwal paling pertama, dimesin yang manapun. Sedangkan $C_{i0\bullet 0\bullet\bullet}$ menyatakan waktu selesai dari semua operasi item *i* pada semua mesin. Dimana $C_{i0\bullet 0\bullet\bullet}$ dapat diartikan item *i* dapat terselesaikan, sebaga dasar pentuan *due date* dengan *customer*.

Adapun fungsi kendala dari permasalahan ini terdiri dari beberapa kendala yang berhubungan dengan ketersediaan mesin dan hubungan ketergantungan pekerjaan pada setiap levelnya.

- Fungsi kendala hubungan kuantitas item dengan kuantitas komponen.

$$n_{ij} = n_{i0} * H_{ij}\left(\prod_{p_{iq}\in Z(p_{ij})} H_{iq}\right), \quad H_{iq} \underset{p_{iq}\in Z(p_{ij})}{\geq} 1 \; ; \forall\, i, j \tag{3.14}$$

  Persamaan (3.17) menyatakan hubungan kuantitas item yang dibuat dengan kuantitas setiap komponennya. H_{ij} menyatakan jumlah komponen ke-*j* yang diperlukan untuk membuat item *i* untuk membuat satu unit induk langsung. Dengan p_{iq} menyatakan item ke-*q* sebagai elemen himpunan $Z(p_{ij})$ dari induk-induk komponen ke-*j* hingga item *i*. persamaan ini untuk menjamin bahwa didalam penentuan jumlah komponen ke-*j* harus memperhatikan struktur produk penyusunannya.

- Fungsi kendala hubungan tingkat cacat (*defect rate*) terhadap jumlah komponen yang diproduksi.

$$n_{ijk(diproduksi)} \geq n_{ij(k+1)(diproduksi)} * (1 + dr_{ijkm\bullet}) \tag{3.15}$$

  Dimana, $n_{ijk+1(diproduksi)} \geq n_{ij(k+1)} * (1 + dr_{ijk+1m\bullet})$, sehingga persamaan (3.18) menyatakan bahwa jumlah komponen ke-*j* dari item *i* pada operasi ke-*k* yang harus diproduksi, pada kondisi suatu item harus

diproses lebih dari satu kali pada mesin yang berbeda, ditentukan oleh jumlah komponen pada operasi ke-k+1 dan *defect rate* (*dr*) untuk semua mesin *m* yang identik yang tersedia dalam tahapan produksi pada operasi ke-$k+1$.

- Fungsi kendala hubungan jumlah komponen atau part dalam *batch* dengan jumlah komponen akhir.

$$\sum_{u=1}^{N} Q_{ij}[u] = n_{ij} \; ; \forall \, i, j \tag{3.16}$$

Persamaan (3.19) menyatakan keseimbangan material, artinya bahwa jumlah komponen atau item yang diproduksi pada seluruh *batch* sebanyak *N batch* sama dengan jumlah komponen atau item bersangkutan, sebelum dilakukan pemecahan kedalam *batch-batch* produksi.

- Fungsi kendala kelayakan waktu memulai penjadwalan

$$B_{ij1\bullet\bullet\bullet}[N] \geq 0 \; ; \forall \, i, j \tag{3.17}$$

Persamaan (3.20) menjamin bahwa waktu dimulai pemprosesan *batch* pertama dari komponen p_{ij} harus lebih besar atau sama dengan nol, yang berarti pekerjaan tersebut layak dikerjakan didalam horizon waktu perencanaan.

- Fungsi kendala hubungan waktu penyelesaian komponen terhadap induk langsungnya.

$$C_{ijhl+1\bullet\bullet} - B_{ij1l\bullet\bullet} \leq 0 \; ; \forall \, i, j, k, l \tag{3.18}$$

Persamaa (3.18) menjamin bahwa pemproses *batch* pertama operasi pertama $k=1$ untuk komponen p_{ij} pada level ke-*l* sebagai induk langsung dari komponen p_{ij} pada level ke-$l+1$, dapat dimulai apabila operasi terakhir $k=h$ komponen p_{ij} telah selesai dikerjakan, disemua mesin.

- Fungsi kendala hubungan antar pemprosesan operasi *batch*.

$$C_{ij(k-1)l\bullet\bullet\bullet}[u] - B_{ijkl\bullet\bullet}[u] \leq 0 \; ; \forall \, i, j, k, l, u \tag{3.19}$$

Persamaan (3.19) menyatakan pemproses *batch* ke-*u* suatu operasi bisa dilakukan bila pemprosesan pada operasi sebelumnya telah selesai dikerjakan.

- Fungsi kendala hubungan pemrosesan antar *batch*.

$$B_{ijklm\bullet}[u] \le B_{ijklm\bullet}[u-1] - \left(Q_{ijkl}[u] * t_{ijkm\bullet}\right) - s_{ijkm\bullet};$$
$$\forall\, i, j, k, l, m, u \qquad (3.20)$$

Persamaan (3.20) menyatakan pemprosesan suatu *batch* ke-*u* harus segera dilakukan bila *batch* ke *u-1* atau batch sebelumnya telah selesai diproses.

- Fungsi kendala untuk menjamin bahwa suatu mesin hanya dapat memproses satu *batch* operasi saja pada waktu tertentu.

$$C_{\bullet\bullet\bullet\bullet mn}[w] - C_{\bullet\bullet\bullet\bullet mn}[y] + \alpha\left(X_{wy}\right) \ge \left(Q_{\bullet\bullet\bullet\bullet}[w] * t_{\bullet\bullet\bullet\bullet mn}\right);$$
$$\forall\, m, n \qquad (3.21)$$

$$C_{\bullet\bullet\bullet\bullet mn}[y] - C_{\bullet\bullet\bullet\bullet mn}[w] + \alpha\left(1 - X_{wy}\right) \ge \left(Q_{\bullet\bullet\bullet\bullet}[y] * t_{\bullet\bullet\bullet\bullet mn}\right);$$
$$\forall\, m, n \qquad (3.22)$$

Persamaan (3.21 dan 3.22) menyatakan waktu penyelesaian *batch* pada posisi *w*, apabila dikurangi dengan waktu penyelesaian *batch* pada posisi berikutnya (posisi *y*) di tambah dengan bilangan positif sangat besar (α) yang lebih besar atau sama dengan waktu diperlukan untuk menyelesaikan jumlah part dalam *batch* pada posisi ke-*w*. kondisi ini berlaku untuk *batch* dengan urutan lebih awal diproses lebih dahulu pada mesin-*m* ke-*n*. apabila urutan pemprosesan dibalik akan berlaku persamaan (3.22).

- Fungsi kendala untuk kondisi variabel.

$$X_{wy} \in \{0,1\},\ Q_{ijkl}[u] > 0, \text{ dan } N_{ijk} \ge 1, \text{integer} \qquad (3.23)$$

Persamaan (3.14) menyatakan nilai X_{wy} adalah 1 jika *batch* operasi ke-*k* diposisi *w* yaitu $L_{[w]}$ mendahului $L_{[y]}$, atau bernilai 0 jika sebaliknya, ukuran *batch* harus bernilai positif, dan jumlah *batch* merupakan bilangan *integer* yang bernilai lebih besar atau sama dengan satu.

Dikarenakan formulasi model penentuan *due* date atau model dua tersebut merupakan model yang kompleks karena banyak variabel yang terkait maka.

Untuk itu dilakukan pendekatan heuristik sama seperti pada model satu. Dengan melakukan pengubahan terhadap beberapa variabel menjadi parameter seperti yang telah dilakukan sebelumnya.

Dimana secara garis besar algoritma yang dikembangkan bekerja dengan tahapan sebagai berikut:

- Mengurutan item-item berdasarkan total waktu *set-up* dan waktu prosesnya secara tidak mengecil.
- Melakukan penentuan jumlah dan ukuran *batch* secara bertahap.
- Melakukan penjadwalan secara *forward* dimulai dari *t=0.*
- Melakukan penggabungan jadwal pada tiap mesin untuk memperoleh S_{Akhir}.
- Menentukan *due date* masing-masing item.

3.4.1 Algoritma Penentuan Due Date Kondisi Statis

Di dalam penyelesaian permasalahan penentuan *due date* memiliki prinsip penyelesaian yang relatif sama dengan permasalahan pemenuhan *due date*, tetapi terdapat berubahan pada aturan penentuan prioritas penjadwalan. Secara garis besar alur langkah-langkah penyelesaian permasalah ditunjukan pada **Gambar 3.7.**

Pada permasalahan penentuan *due date* pada kondisi statis diusulkan algoritma sebagai berikut:

Pesanan-pesanan yang telah ada pada $t = 0$ dijadwalkan mengikuti aturan:

Langkah 0. Tentukan $P(p_{i0})$ yaitu set dari item-item yang akan diproduksi.

Langkah 1. Urutkan item-item pada $P(p_{i0})$ sesuai dengan urutan jumlah total waktu *set up* dan operasi dengan mempertimbangakan tingkat cacat pada setiap proses secara menaik dan nyatakan sebagai $P'(p_{i0})$. Jika produk akhir $P(p_{i0})$ diproses pada dua mesin atau lebih, maka kuantitas pada posisi *k* ditentukan dari kuantitas pada posisi *k+1*.

Langkah 2. Tetapkan item yang menempati urutan pertama pada $P'(p_{i0})$ sebagai item $\alpha = 1$ (dengan indeks $\alpha = 1, 2, ..., \beta$).

Langkah 3. Buat struktur dari item bersangkutan dan gambarkan *digraph* dari produk akhir dan komponen-komponennya. Tiap tahapan pada

digraph dinyatakan dengan *level*, dimana penomoran *level* produk dimulai dengan menyatakan *level 0* untuk produk akhir, kemudian nomor *level* bertambah satu pada setiap langkah mundur hingga mencapai komponen pada *level* terendah (L).

Langkah 4. Tentukan jenis komponen yang akan diproduksi dan jumlah mesin yang digunakan $(p_{ij}$ dan $m)$.

Langkah 5. Untuk produk akhir, tentukan jumlah produk akhir yang harus dibuat dengan mengakomodasi tingkat cacat dari tiap proses, operasi-operasi untuk produk akhir, waktu *set-up* dan waktu proses operasi-operasi tersebut di masing-masing mesin untuk produk akhir $(n_{i0}, o_{i0klm}, s_{i0klm}, dan\, t_{i0klm})$. Jika produk akhir diproses pada dua mesin atau lebih, maka kuantitas pada posisi *k* ditentukan dari kuantitas pada posisi *k+1*.

Langkah 6. Untuk komponen-komponen produk akhir, tentukan induk induk komponen bersangkutan hingga produk akhir dan banyaknya komponen bersangkutan yang dibutuhkan untuk membuat satu unit induk langsung $(Z(p_{ij})$ dan $H_{ij})$.

Langkah 7. Untuk tiap komponen, tentukan jumlah komponen yang harus diproduksi dengan mengakomodasi tingkat cacat tiap proses, operasi-operasi untuk komponen bersangkutan, dan waktu *set-up* dan waktu proses operasi-operasi tersebut di masing-masing mesin $(n_{ij}, o_{ijklm}, s_{ijklm}, \text{dan } t_{ijklm})$ dengan $n_{ij} = n_{i0} * H_{ij}\left(\prod_{p_{ik} \in Z(p_{ij})} H_{ik} \right)$ dimana $\prod_{p_{ik} \in Z(p_{ij})} H_{ik} \geq 1$. Dengan menggunakan prinsip *withdrawl kanban* dimana kuantitas pada urutan *k* ditentukan pada kebutuhan diurutan *k+1*.

Langkah 8. Karena pendekatan yang digunakan adalah pendekatan maju (*forward approach*), maka tahapan pemrosesan mengikuti aturan *maximum level first* (MaLF), dimulai dari *level l* atau produk akhir.

Langkah 9. Gunakan **[Sub-Algoritma Penentuan Jumlah dan Ukuran *Batch*]**. Lanjutkan ke *Langkah 10*.

Langkah 10. Set $l = l - 1$.

- Jika $l \geq 0$, kembali ke *Langkah 9*.
- Jika $l < 0$, lanjutkan ke *Langkah 11*.

Langkah 11. Set $\alpha = \alpha + 1$.

- Jika $\alpha \leq \beta$, kembali ke *Langkah 3*.
- Jika $\alpha > \beta$, lanjutkan ke *Langkah 12*.

Langkah 12. Penjadwalan secara *forward* dimulai dari jumlah total waktu *set-up* dan operasi terkecil.

Langkah 13. Untuk masing-masing mesin, tentukan $O(o_{ijklm})$ yaitu set operasi-operasi dari komponen-komponen yang akan diproses pada mesin bersangkutan dan akomodasi ketersediaan dari mesin tersebut.

Langkah 14. Untuk tiap mesin, lakukan *Langkah 15 - 28*.

Langkah 15.Kelompokkan komponen yang sejenis dalam satu *batch*, $N_{ij} = 1$, sesuai dengan mesinnya.

Langkah 16. Cari operasi-operasi paling hulu yang belum dijadwalkan.

Langkah 17. Apakah pada saat t tertentu, terdapat lebih dari satu operasi yang membutuhkan mesin bersangkutan ?

- Jika ya, lanjutkan ke *Langkah 18*.
- Jika tidak, lakukan pemecahan *batch* untuk komponen tersebut dengan menggunakan hasil yang didapatkan sebelumnya dari **[Sub-Algoritma Penentuan Jumlah dan Ukuran *Batch*]** dan jadwalkan *batch-batch* komponen tersebut dengan urutan berdasarkan Teorema 1. Lanjutkan ke *Langkah 28*.

Langkah 18. Apakah indeks produk akhir dari komponen untuk operasi operasi itu berbeda ?

- Jika ya, lanjutkan ke *Langkah 19*.
- Jika tidak, lanjutkan ke *Langkah 21*.

Langkah 19. Urutkan operasi-operasi berdasarkan urutan produk akhir pada $P'(p_{i0})$. Beri indeks $\alpha = 1, 2, ..., \beta$.

Langkah 20. Mulai dari produk pada level terakhir yang berada pada urutan pertama dan set $\alpha = 1$.

Langkah 21. Apakah terdapat lebih dari satu operasi yang membutuhkan mesin bersangkutan ?

- Jika ya, lanjutkan ke *Langkah 22*.
- Jika tidak, lakukan pemecahan *batch* untuk komponen tersebut dengan menggunakan hasil yang didapatkan sebelumnya dari **[Sub-Algoritma Penentuan Jumlah dan Ukuran *Batch*]** dan jadwalkan *batch-batch* komponen tersebut dengan urutan berdasarkan Teorema 1. Lanjutkan ke *Langkah 28*.

Langkah 22. Kelompokkan komponen yang sejenis dalam satu *batch*, $N_{ij} = 1$ sesuai dengan mesinya dan dengan Teorema 2 tentukan urutan yang menghasilkan total waktu tinggal aktual yang paling minimum. Beri indeks $x = 1,2,...,y$.

Langkah 23. Pilih *batch* untuk komponen yang berada paling dekat dengan *due-date*, nyatakan sebagai *batch* komponen $x = 1$.

Langkah 24.Lakukan pemecahan *batch* untuk komponen x tersebut dengan menggunakan hasil yang didapatkan sebelumnya dari **[Sub-Algoritma Penentuan Jumlah dan Ukuran *Batch*]**.

Langkah 25.Jadwalkan *batch-batch* komponen tersebut dengan urutan berdasarkan Teorema 1.

Langkah 26. Set $x = x + 1$.

- Jika $x \le y$, kembali ke *Langkah 24*.
- Jika $x > y$, lanjutkan ke *Langkah 27*.

Langkah 27. Set $\alpha = \alpha + 1$.

- Jika $\alpha \le \beta$, kembali ke *Langkah 21*.
- Jika $\alpha > \beta$, lanjutkan ke *Langkah 28*.

Langkah 28.Ulangi *Langkah 16* sampai seluruh operasi pada mesin bersangkutan telah dijadwalkan. Jika semua operasi pada mesin bersangkutan telah terjadwal ulangi *Langkah 14*.

Langkah 29.Penggabungan jadwal pada tiap-tiap mesin untuk membentuk S_{akhir}. Untuk tiap-tiap mesin, lakukan *Langkah 30 - 38*.

Langkah 30.Tempatkan operasi pertama untuk produk pertama sedemikian sehingga saat mulai operasi bersangkutan tepat pada *t=0* atau sedini mungkin.

Langkah 31.Tentukan $O'(o_{ijklm})$ yaitu set operasi-operasi dari komponen komponen sesuai dengan urutan pemrosesan pada mesin bersangkutan (kecuali operasi yang telah ditempatkan pada *Langkah 30*). Beri indeks $x = 1,2,...,y$.

Langkah 32.Mulai dengan operasi yang menempati urutan pertama pada $O'(o_{ijklm})$ dan nyatakan sebagai operasi $x = 1$.

Langkah 33. Apakah pada saat t tertentu, terdapat operasi yang telah terjadwal ?

- Jika ya, lanjutkan ke *Langkah 34*.
- Jika tidak, jadwalkan *batch-batch* operasi x. Lanjutkan ke *Langkah 38*.

Langkah 34.Apakah indeks produk akhir dari komponen untuk operasi x sama dengan indeks untuk operasi terjadwal ?

- Jika ya, lanjutkan ke *Langkah 35*.
- Jika tidak, lanjutkan ke *Langkah 37*.

Langkah 35.Apakah indeks komponen untuk operasi x sama dengan indeks untuk operasi terjadwal ?

- Jika ya, jadwalkan *batch-batch* operasi x di belakang (mengikuti) *batch-batch* operasi yang telah terjadwal tersebut. Lanjutkan ke *Langkah 38*.
- Jika tidak, lanjutkan ke *Langkah 36*.

Langkah 36.Apakah komponen untuk operasi x berada pada *level* yang sama dengan komponen untuk operasi terjadwal ?

- Jika ya, lanjutkan ke *Langkah 37*.
- Jika tidak, jadwalkan *batch-batch* operasi x di belakang (mengikuti) *batch-batch* operasi yang telah terjadwal tersebut. Lanjutkan ke *Langkah 38*.

Langkah 37.Evaluasi posisi penyisipan yang mungkin untuk *batch-batch* operasi x yaitu di depan (mendahului) atau di belakang (mengikuti) *batch-batch* operasi yang telah terjadwal tersebut. Pilih posisi penyisipan

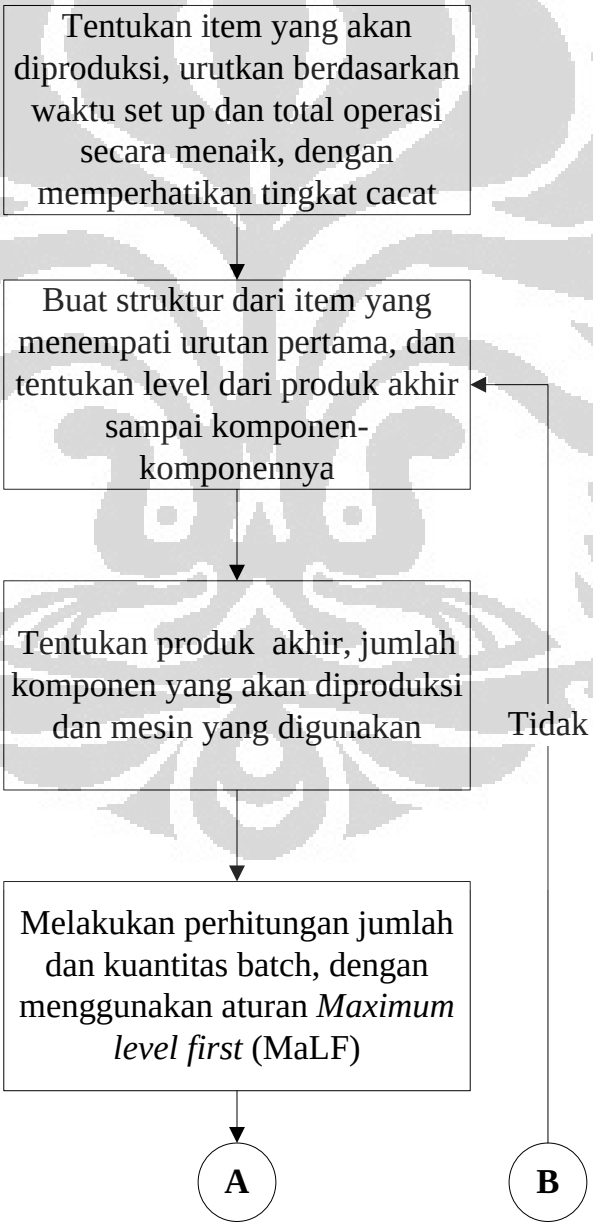
yang menghasilkan total waktu tinggal aktual paling minimum, lalu jadwalkan *batch-batch* operasi x. Lanjutkan ke *Langkah 38*.

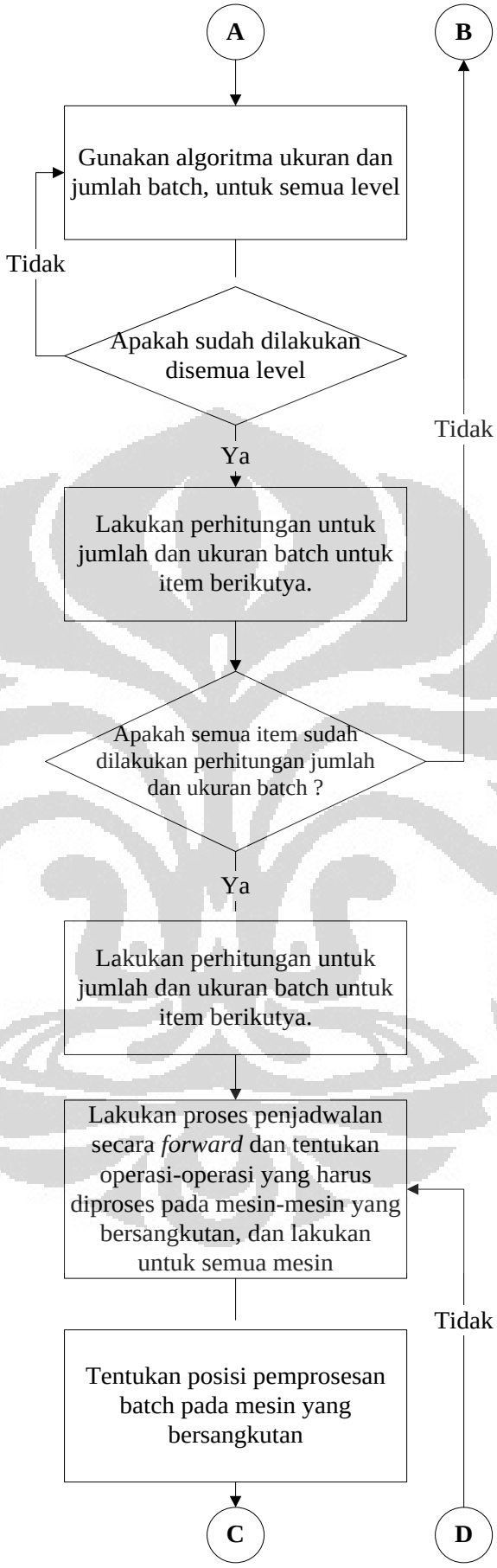
Langkah 38. Set $x = x + 1$.

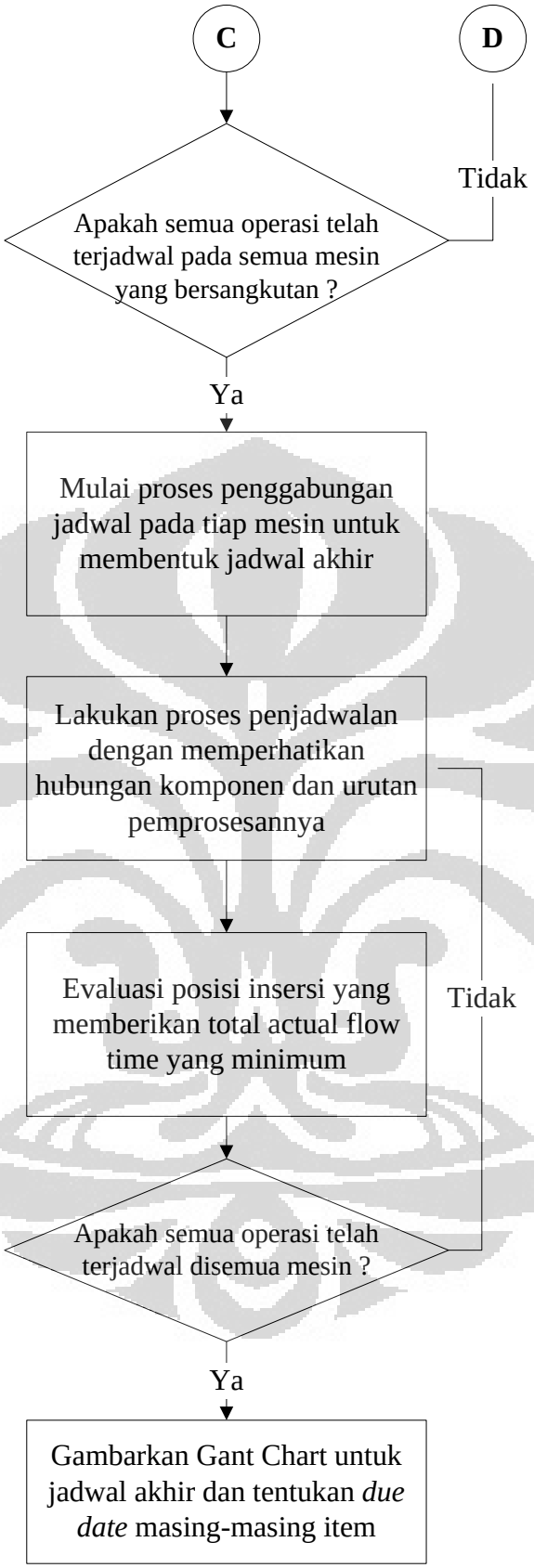
- Jika $x \leq y$, kembali ke *Langkah 33*.
- Jika $x > y$, lanjutkan ke *Langkah 39*.

Langkah 39.Gambarkan *Gantt-Chart* untuk jadwal yang terbentuk (S_{akhir}) dan tentukan due date dari masing-masing item.

Sub algoritma penentuan jumlah dan ukuran *batch* yang digunakan, sama seperti algoritma pada **Sub Bab 3.2.2.**







Gambar 3.6 Diagram Alir Penentuan *Due Date* Job Shop Statis

3.4.2 Penentuan *Due Date* pada Job Shop Dinamis

Jika terjadi kedatangan pesanan-pesanan baru pada saat produksi sedang berjalan, maka digunakan model dua yaitu algoritma untuk menyelesaikan masalah penentuan *due date Job Shop* kondisi dinamis. Algoritma merupakan sekumpulan langkah-langkah yang dilakukan dengan adanya kedatangan pesanan baru pada saat T=*A*, alur penyelesaikan pada kondisi dinamis ditunjukan apda **gambar 3.7** , dengan langakah-langkah detail penyelesaian sebagai berikut.

Langkah 0. Tentukan jumlah masing-masing pesanan baru tersebut. Nyatakan $I(p_{i0})$ yaitu set dari pesanan-pesanan baru dengan mengurutan total waktu *set up* dengan mempertimbangakan tingkat cacat pada setiap proses secara menaik dan beri indeks $\gamma = 1,2,...,\sigma$.

Langkah 1. Mulai dengan pesanan baru yang menempati urutan pertama pada $I(p_{i0})$ dan nyatakan sebagai pesanan $\gamma = 1$.

Langkah 2. Susun jadwal terpisah untuk pesanan baru dengan menggunakan **[Algoritma Penentuan *Due Date Job Shop* Statis]** untuk kasus item tunggal.

Langkah 3. Pada jadwal yang telah ada akan dilakukan penyisipan operasi operasi pesanan baru pada posisi yang memungkinkan secara *forward*.

Langkah 4. Untuk tiap mesin, lakukan *Langkah 5 - 13*.

Langkah 5. Untuk masing-masing mesin tersebut, tentukan $I(o_{ijklm})$ yaitu set operasi-operasi dari komponen-komponen pesanan baru yang akan dipenyisipankan pada mesin bersangkutan, serta akomodasi ketersediaan mesin tersebut, sesuai dengan urutan pada jadwal hasil *Langkah* 2. Beri indeks $x = 1,2,...,y$.

Langkah 6. Periksa apakah ada *batch* yang belum selesai diproses.

- Jika ya, lanjutkan ke *Langkah 7*.
- Jika tidak, lanjutkan ke *Langkah 9*.

Langkah 7. Periksa apakah *batch* yang belum selesai diproses tersebut adalah *batch* nomor 1.

- Jika ya, maka selesaikan pemrosesan *batch* tersebut. Lanjutkan ke *Langkah 8*.
- Jika tidak, selesaikan pemrosesan sampai *batch* nomor 1. Kemudian lanjutkan ke *Langkah 8*.

Langkah 8. Simpan saat selesai pemrosesan *batch* dan nyatakan sebagai $(T_{av})_{mn}$, yaitu titik waktu dimana mesin m ke *n* tersedia untuk melakukan pemprosesan operasi-operasi berikutnya.

Langkah 9. Mulai dengan operasi yang menempati urutan pertama pada $I(o_{ijklm})$ dan nyatakan sebagai operasi $x=1$.

Langkah 10. Apakah pada t tertentu, terdapat operasi yang telah terjadwal ?

- Jika ya, lanjutkan ke *Langkah 11*.
- Jika tidak, jadwalkan operasi x. Simpan saat mulai operasi tersebut dan nyatakan sebagai t_x. Lanjutkan ke *Langkah 13*.

Langkah 11. Evaluasi posisi penyisipan yang mungkin untuk operasi x yaitu di depan (mendahului) atau di belakang (mengikuti) operasi yang telah terjadwal tersebut. Pilih posisi penyisipan yang menghasilkan total waktu tinggal aktual yang paling minimum.

Catatan: penyisipan tidak diantara *batch-batch* operasi dari komponen yang sama.

Langkah 12. Simpan saat mulai operasi x dan nyatakan sebagai t_x.

Langkah 13. Set $x = x+1$.

Catatan: posisi penyisipan untuk operasi-operasi ini harus memiliki saat mulai yang lebih besar dari operasi pada urutan sebelumnya atau $(t_{x+1} > t_x)$.

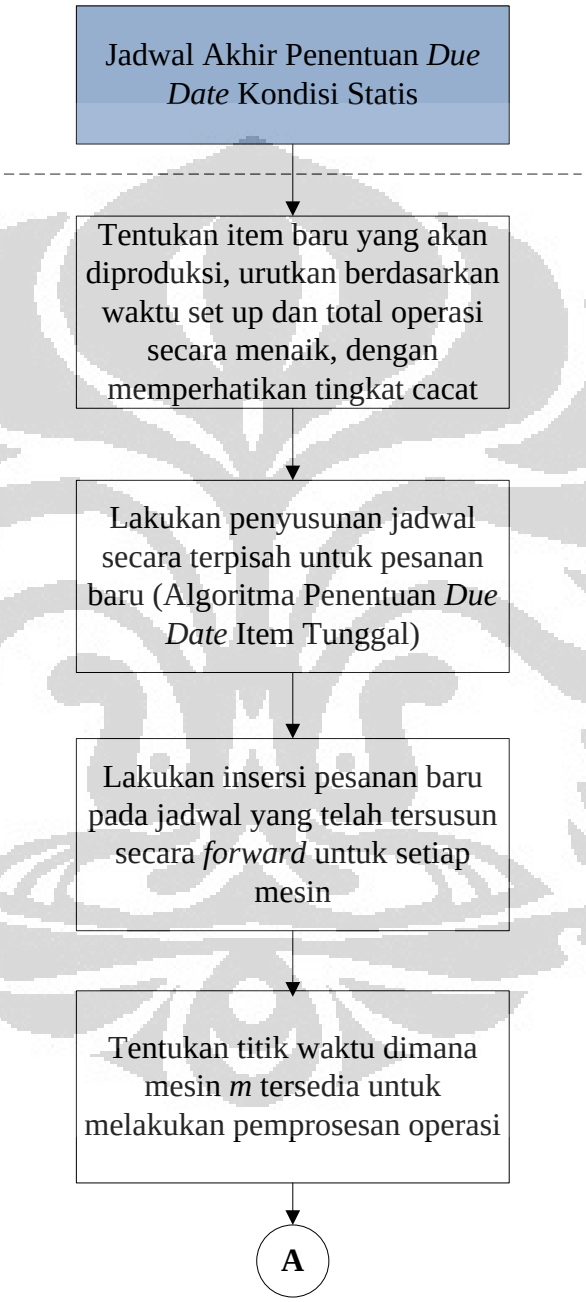
- Jika $x \le y$, kembali ke *Langkah 10*.
- Jika $x > y$, lanjutkan ke *Langkah 14*.

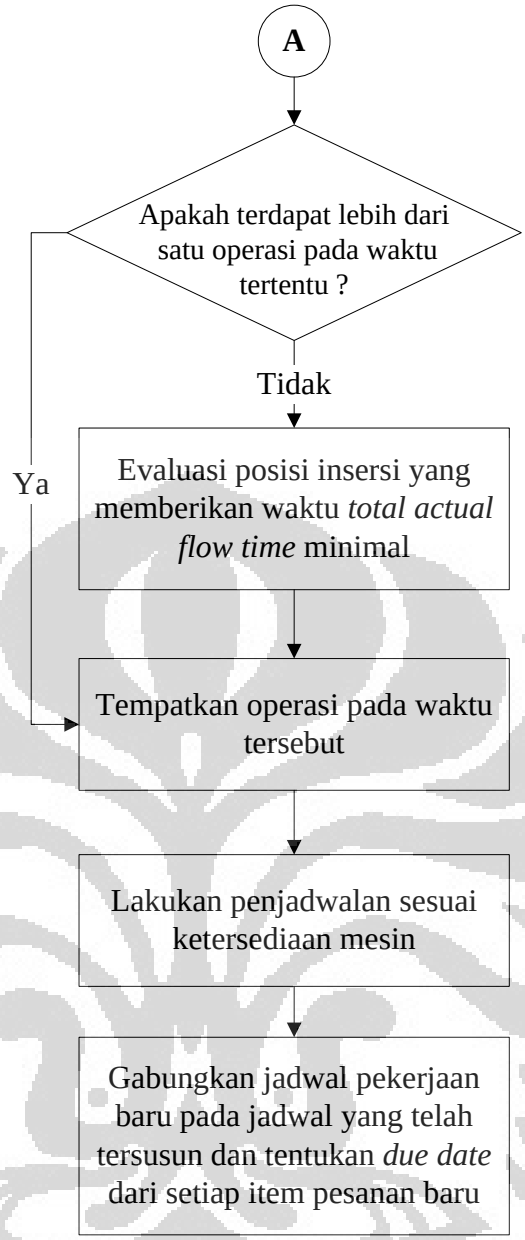
Langkah 14. Gabungkan jadwal pada tiap-tiap mesin untuk membentuk S_{akhir}.

Langkah 15. Set $\gamma = \gamma + 1$.

- Jika $\gamma \leq \sigma$, kembali ke *Langkah 2*.
- Jika $\gamma > \sigma$, lanjutkan ke *Langkah 16*.

Langkah 16. Tampilkan S_{akhir} dan tentukan due date dari setiap item pesanan baru.





Gambar 3.7 Diagram Alir Penentuan *Due Date* Job Shop Dinamis

3.4.2.1 Algoritma Penentuan *Due Date* pada *Job Shop* Statis Item Tunggal

Proses menyelesaian permasalahan penentuan *due date* pada *job shop* yang memproduksi item tunggal, ditunjukan pada **gambar 3.8**. Pada kondisi di asumsikan pesanan yang telah ada pada $t = 0$ dijadwalkan mengikuti langkah-langkah, sebagai berikut.

Langkah 0. Buat struktur dari item bersangkutan dan gambarkan *digraph* dari produk akhir dan komponen-komponennya. Tiap tahapan pada *digraph* dinyatakan dengan *level*, dimana penomoran *level* produk dimulai dengan menyatakan *level 0* untuk produk akhir, kemudian

nomor *level* bertambah satu pada setiap langkah mundur hingga mencapai komponen pada *level* terendah (L).

Langkah 1. Tentukan jenis komponen yang akan diproduksi dan jumlah mesin yang digunakan $(p_{ij}$ dan $m)$.

Langkah 2. Untuk produk akhir, tentukan jumlah produk akhir yang harus dibuat dengan mengakomodasi tingkat cacat dari tiap proses, operasi-operasi untuk produk akhir, waktu *set-up* dan waktu proses operasi-operasi tersebut di masing-masing mesin untuk produk akhir $(n_{i0}, o_{i0k0m}, s_{i0km},$ dan $t_{i0km})$.

Langkah 3. Untuk komponen-komponen produk akhir, tentukan induk induk komponen bersangkutan hingga produk akhir dan banyaknya komponen bersangkutan yang dibutuhkan untuk membuat satu unit induk langsung $(Z(p_{ij})$ dan $H_{ij})$.

Langkah 4. Untuk tiap komponen, tentukan jumlah komponen yang harus diproduksi dengan mengakomodasi tingkat cacat tiap proses, operasi-operasi untuk komponen bersangkutan, dan waktu *set-up* dan waktu proses operasi-operasi tersebut di masing-masing mesin $(n_{ij}, o_{ijklm}, s_{ijklm},$ dan $t_{ijklm})$ dengan

$$n_{ij} = n_{i0} * H_{ij}\left(\prod_{p_{iq} \in Z(p_{ij})} H_{iq}\right)$$ dimana $$H_{iq} \geq 1$$ $$p_{iq} \in Z(p_{ij})$$. Dengan menggunakan prinsip *withdrawl kanban* dimana kebutuhan pada urutan *k* ditentukan berdasarkan kebutuhan pada *k+1*.

Langkah 5. Karena pendekatan yang digunakan adalah pendekatan maju (*forward approach*), maka tahapan pemrosesan mengikuti aturan *maximum level first* (MaLF), dimulai dari *level l* atau produk awal.

Langkah 6. Gunakan **[Sub-Algoritma Penentuan Jumlah dan Ukuran *Batch*]**. Lanjutkan ke *Langkah 7*.

Langkah 7. Tentukan $M(p_{il})$ yaitu set dari mesin-mesin yang digunakan untuk pemrosesan komponen-komponen yang berada pada *level l* struktur produk akhir i.

Langkah 8. Untuk masing-masing mesin tersebut, tentukan pula $O(o_{ijklm})$ yaitu set operasi-operasi dari komponen-komponen yang akan diproses pada mesin bersangkutan.

Langkah 9. Untuk tiap mesin, lakukan *Langkah 10 - 17*.

Langkah 10.Cari operasi-operasi pada urutan pertama atau paling hulu yang belum dijadwalkan.

Langkah 11. Apakah pada saat t tertentu, terdapat lebih dari satu operasi yang membutuhkan mesin bersangkutan ?

- Jika ya, lanjutkan ke *Langkah 12*.
- Jika tidak, lakukan pemecahan *batch* untuk komponen tersebut dengan menggunakan hasil yang didapatkan sebelumnya dari **[Sub-Algoritma Penentuan Jumlah dan Ukuran *Batch*]** dan jadwalkan *batch-batch* komponen tersebut dengan urutan berdasarkan Teorema 1. Lanjutkan ke *Langkah 17*.

Langkah 12. Kelompokkan komponen yang sejenis dalam satu *batch*, $N_{ij} = 1$, sesuai dengan mesinnya dan dengan Teorema 2 tentukan urutan yang menghasilkan total waktu tinggal aktual yang paling minimum. Beri indeks $x = 1,2,...,y$.

Langkah 13. Pilih *batch* untuk komponen yang berada paling dekat dengan *t=0*, nyatakan sebagai *batch* komponen $x = 1$.

Langkah 14. Lakukan pemecahan *batch* untuk komponen x tersebut dengan menggunakan hasil yang didapatkan sebelumnya dari **[Sub-Algoritma Penentuan Jumlah dan Ukuran *Batch*]**.

Langkah 15.Jadwalkan *batch-batch* komponen tersebut dengan urutan berdasarkan Teorema 1.

Langkah 16. Set $x = x + 1$.

- Jika $x \le y$, kembali ke *Langkah 14*.
- Jika $x > y$, lanjutkan ke *Langkah 17*.

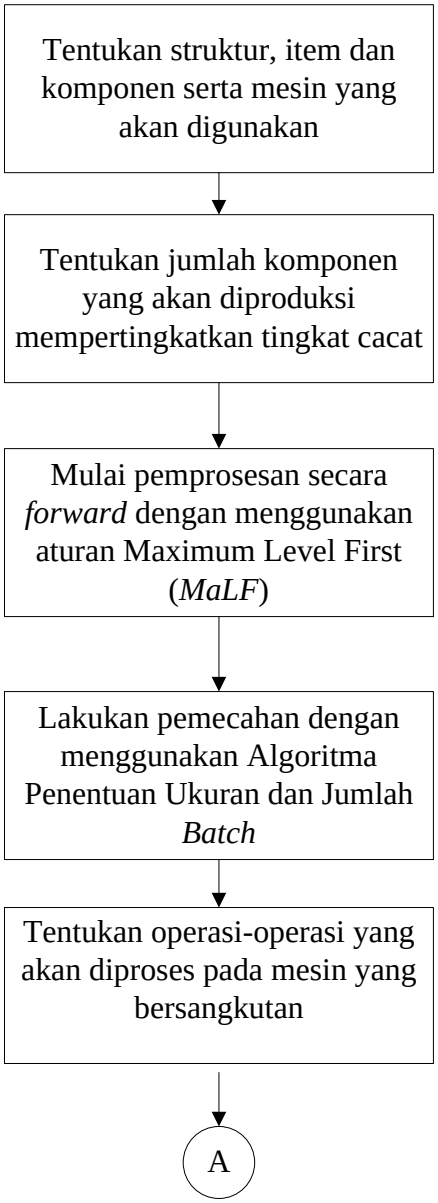
Langkah 17. Ulangi *Langkah 10* sampai seluruh operasi pada mesin bersangkutan telah dijadwalkan.

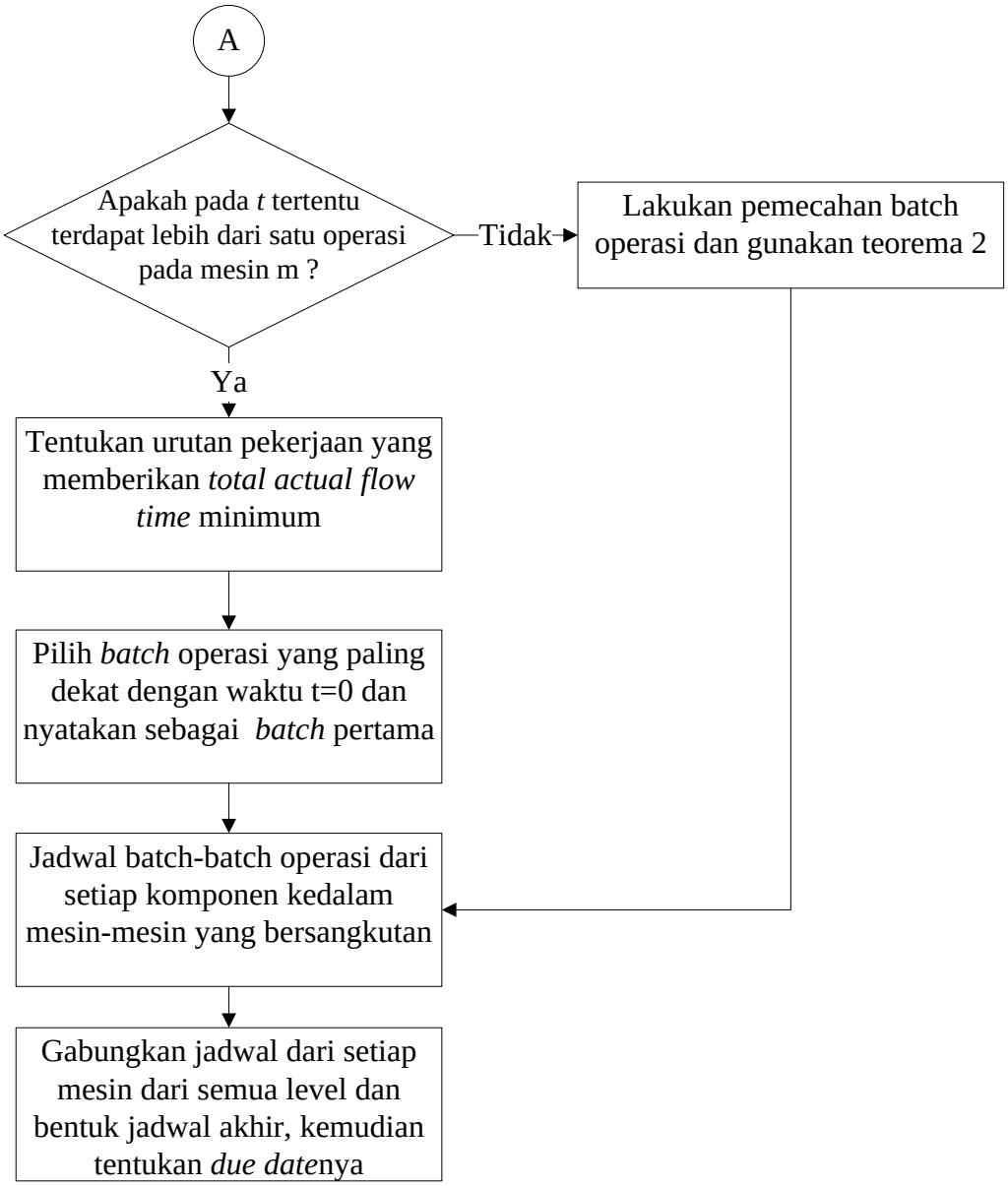
Langkah 18. Gabungkan jadwal pada tiap-tiap mesin untuk membentuk S_l yaitu jadwal parsial untuk *level* l struktur produk akhir i.

Langkah 19. Set $l = l - 1$.

- Jika $l \geq 0$, kembali ke *Langkah 6*.
- Jika $l < 0$, lanjutkan ke *Langkah 20*.

Langkah 20.Gabungkan jadwal pada tiap-tiap *level* untuk membentuk S_{akhir} dimana $S_{akhir} = \{S_0, S_1, ..., S_l\}$ secara *forward*.





Gambar 3.8 Diagram Alir Penentuan *Due Date* Item Tunggal

BAB IV
PENGUJIAN DAN ANALISA MODEL

4.1 Verifikasi dan Validasi Model

Untuk memastikan algoritma yang dikembangkan dapat bekerja untuk menyelesaikan permasalahan sesuai model yang telah diformulasikan. Didalam proses verifikasi dan validasi ini dilakukan dengan menggunakan hipotesis (*Hypothetical*) data dalam ukuran kecil dan ukuran besar. Data dalam ukuran kecil akan diselesaikan secara perhitungan manual, agar dapat dapat dilihat langkah demi langkah dari algoritma yang dikembangkan bekerja dalam menyelesaikan permasalahan. Pengujian algoritma pada data yang lebih besar akan dilakukan dengan menggunakan program komputer yang dikembangkan.

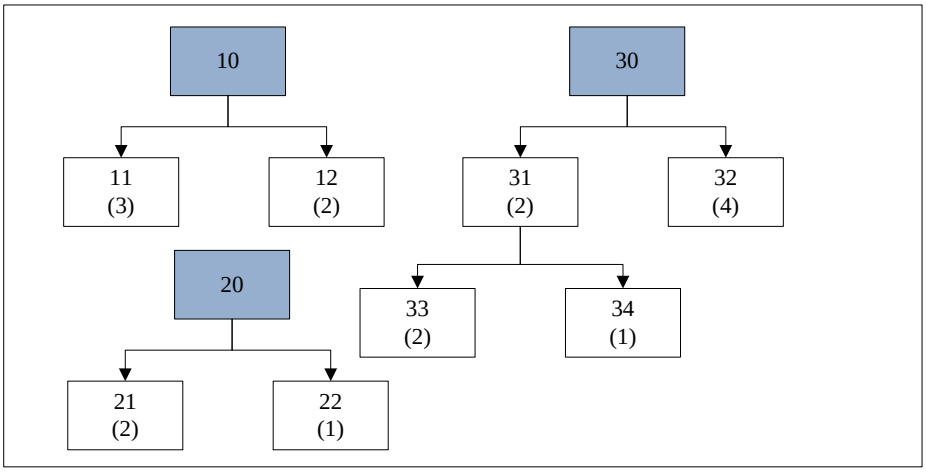
Data hipotesis yang akan digunakan dalam verifikasi dan validasi model secara manual terdiri dari tiga item produk akhir yang diproduksi, dengan masing-masing item memiliki dua level (yaitu level 0 dan level 1 serta satu item memiliki level 2) dan diproses lebih dari satu mesin. Di mana setiap jenis mesin dapat memiliki lebih dari satu mesin identik. Kondisi ini, untuk menjamin bahwa algoritma yang dikembangkan dapat menyelesaikan permasalahan pada lingkungan *job shop* yang memproduksi multi item dan berstruktur multi level, dengan mesin *paralel*.

4.1.1 Set Data Pengujian Kondisi Statis

Terdapa 3 (tiga) buah produk atau item yang diproduksi, setiap produk tersebut terdiri dari (2) dua buah komponen dan lebih dengan kebutuhan setiap komponen untuk membentuk produk akhir berbeda-beda seperti pada Gambar 4.1. Dalam proses produksi digunakan 4 (empat) buah mesin. Adapun data detailnya sebagai berikut.

Tabel 4.1 Data Mesin Tersedia

No	Mesin	Jumlah (Unit)
1	1	3
2	2	1
3	3	3
4	4	2



Gambar 4.1 Struktur Setiap Produk

Adapun kuantitas yang harus diproduksi dan *due date* dari masing-masing pekerjaan seperti ditunjukan pada Tabel 4.2, berikut ini.

Tabel 4.2 Data Item, Kuantitas dan *Due Date*

P_{i0}	**Kuatitas (unit)**	***Due-date* (menit)***
10	10	3400
20	15	3250
30	20	3300

***** untuk permasalahan pemenuhan *due date*

Setiap item dan komponen dilakukan pemprosesan pada setiap meisn sesuai dengan *routing* atau urutan pemprosesnya, data waktu *set up*, waktu proses dan tingkat cacat seperti ditunjukan pada Tabel 4.3, seperti ditunjukan berikut ini.

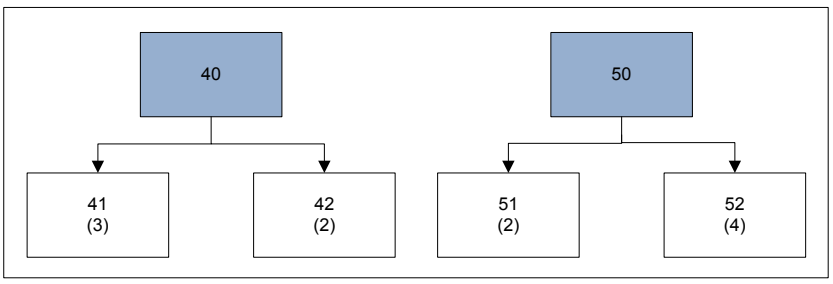
Tabel 4.3 Waktu Proses, Set Up dan Tingkat Cacat

P_{ij}	**Operation**	**Mesin**	***Set-up time* (menit)**	**Waktu proses (menit)**	**Tingkat Cacat (%)**
10	1	3	30	10	4
20	1	1	40	15	5
30	1	2	15	5	8
11	1	4	10	5	6
	2	3	20	5	5
12	1	2	25	10	10
	2	1	45	15	4

P_{ij}	Operation	Mesin	*Set-up time* (menit)	Waktu proses (menit)	Tingkat Cacat (%)
21	1	3	15	5	8
	2	2	45	15	6
22	1	1	100	20	5
	2	4	50	15	7
31	1	3	25	5	8
32	1	1	30	10	6
33	1	3	35	5	5
	2	4	50	10	10
34	1	4	45	5	5
	2	2	60	15	8

4.1.2 Set Data Pengujian Kondisi Dinamis

Pada saat kegiatan produksi sudah berlangsung terjadi kedatangan pesanan baru pada T= 2100 (menit) untuk permasalahan pemenuhan *due date* dan pada T= 1100 untuk permasalahan pemenuhan *due date*, dimana terjadi dua buah kedatangan pesanan baru dengan struktur produk seperti pada Gambar 4.2, sebagai berikut.



Gambar 4.2 Struktur Pesanan Baru

Adapun data mengenai kuantitas yang harus diproduksi, *due date*, urutan pemprosesan, waktu proses, waktu set up dan tingkat cacat seperti ditunjukan pada Tabel 4.4 dan Tabel 4.5, sebagai berikut.

Tabel 4.4. Data Item, Kuantitas dan *Due Date* Pesanan Baru

P_{i0}	**Kuantitas**	***Due date* (menit)***
40	15	3875
50	25	3950

***** digunakan untuk permasalahan pemenuhan due date

Tabel 4.5 Waktu Proses, Set Up dan Tingkat Cacat

P_{ij}	**Operasi**	**Mesin**	**Waktu *set up* (menit)**	**Waktu proses (menit)**	**Defect Rate (%)**
40	1	3	25	10	6
50	1	2	15	5	4
41	1	4	10	1	6
	2	3	20	5	5
42	1	3	25	10	10
	2	1	15	15	4
51	1	4	20	5	8
52	1	1	15	1	6

4.1.3 Hasil Pengujian Kondisi Statis Pemenuhan *Due- Date*.

Dengan menggunakan algoritma model satu yaitu algoritma pemenuhan due date untuk penyelesaian model statis maka diperoleh jadwal akhir untuk data kondisi statis sebagaimana ditunjukan pada Gambar 4.3. Untuk Detail perhitungan dapat dilihat pada lampiran 1.

Berdasarkan hasil perhitungan yang telah dilakukan, maka diketahui kapan waktu mulai pemprosesan operasi pertama setiap pekerjaan dimulai, agar dapat diselesaikan tepat pada *due date*-nya, seperti tujuan dari penelitian ini. Di mana dalam penelitian ini tidak diperkenankan terjadinya keterlambatan yang disebabkan oleh terlewatinya waktu penyelesaian pekerjaan yang dinyatakan dalam *due date*.

Dari jadwal akhir yang terbentuk dan perhitungan yang telah dilakukan, maka diketahui masing-masing pekerjaan dimulai pada waktu sebagai berikut.

Tabel 4.6 Saat Mulai Pengerjaan Masing-masing Item

P_{i0}	**10**	**20**	**30**
Saat mulai menit ke-	2915	2373	1553.75

Sehingga diperpoleh *total actual flow time* (TF) sebesar **3108.25** menit. Dari Tabel 4.6 terlihat bahwa semua pekerjaan layak untuk dikerjakan karena waktu mulai pemprosesan *batch* pertama lebih besar dari nol untuk semua item, dimana untuk item satu (P_{10}) harus dimulai pada t=2915, supaya item tersebut dapat terselesaikan tepat pada *due date*nya. Sedangkan untuk P_{20} dan P_{30} masing-masing pada $t=2373$ dan $t=1553.75$. Dengan waktu mulai dan selesai, dari setiap operasi dari semua item, seperti ditunjukan pada tabel 4.7, sebagai berikut.

Tabel 4.7 Waktu Mulai dan Selesai Setiap Operasi (menit)

O_{ijklm}	S_{ijklmn}	C_{ijklmn}	**Mesin**
O_{101031}	3290	3400	m_{31}
O_{111141}	3082.33	3120.33	m_{41}
O_{121121}	2915	3175	m_{21}
O_{122112}	2945	3290	m_{12}
O_{112132}	3085	3260	m_{31}
O_{201011}	3010	3250	m_{11}
O_{211131}	2373	2558	m_{31}
O_{221111}	2380.05	2855.05	m_{11}
O_{212121}	2380	2890	m_{21}

O_{ijklm}	S_{ijklmn}	C_{ijklmn}	**Mesin**
O_{222141}	2740	3010	m_{41}
O_{301021}	3190	3300	m_{21}
O_{311131}	2825	3065	m_{31}
O_{321113}	2250	3190	m_{13}
O_{331232}	1624.15	2184.15	m_{31}
O_{341241}	1553.75	1828.75	m_{42}
O_{332241}	1630	2690	m_{41}
O_{342221}	1555	2335	m_{21}

Berdasarkan dari hasil perhitungan, mesin m_{32} tidak digunakan didalam pemprosesan penyelesaian pekerjaan. Hal ini, dikarenakan semua pekerjaan dapat terselesaikan dengan menggunakan mesin m_{31}, dengan memberikan hasil *TF* yang sama besarnya. Algoritma yang dikembangkan akan melakukan perubahan terhadap mesin yang akan digunakan dari m_{31} ke m_{32}, apabila setelah dilakukan proses penyisipan disemua posisi yang mungkin tidak memenuhi syarat hubungan ketergantungan antar proses dan dengan induk komponennya.

4.1.4 Hasil Pengujian Kondisi Dinamis Pemenuhan *Due- Date*.

Dengan menggunakan algoritma pada model dua yaitu algoritma untuk penyelesaian pemenuhan *due date* kondisi dinamis. Maka diperoleh *Gantt Chart* akhir seperti pada Gambar 4.4. Untuk detail perhitungan ditunjukan pada Lampiran 2.

Berdasarkan hasil perhitungan yang telah dilakukan, untuk kondisi dinamis dimana terjadi kedatangan pesanana baru pada saat T=2100 yaitu sebanyak dua buah pesanan, yaotu p_{40} dan p_{50}, berdasarkan dari hasil perhitungan yang dilakukan kedua pesanan baru ini dapat diterima. Perhitungan pada kondisi dinamis, terlebih dahulu melakukan evaluasi terhadap semua operasi dari masing-masing item yang akan dalam proses penyelesaian ketika terjadi kedatangan pesanan baru. Dari jadwal akhir untuk kondisi dinamis maka diketahui bahwa pemprosesan dari masing-masing pesanan baru, sebagai berikut:

Tabel 4.8 Saat Mulai Pengerjaan Masing-masing Pesanan Baru

P_{i0}	**4**	**5**
Saat mulai menit ke-	3060	3535

Sehingga diperpoleh *total actual flow time* (TF) yang baru sebesar **4194.05** menit. Dari Tabel 4.9 terlihat waktu mulai dan selesai dari semua operasi dari setiap item, setelah terjadi kedatangan pesanan item baru. Semua operasi dari setiap item yang dalam pemprosesan tidak mengalami perubahan kecuali operasi yang belum dilakukan pemprosesan pada waktu terjadi kedatangan pesanan baru T=2100.

4.1.5 Hasil Pengujian Kondisi Statis Penentuan *Due- Date*

Dengan menggunakan algoritma penenuan *due date* pada model dua. Diperoleh waktu mulai operasi dan waktu selesai dapat pada ditentukan *due date* dari masing-masing item seperti ditunjukan pada Tabel 4.10., detail perhitungaan pada Lampiran 3. Adapun *Gantt Chart* dari setiap operasi seperti ditunjukan pada Gambar 4.5.

Tabel 4.9 Waktu Mulai dan Selesai Operasi Kondisi Dinamis (menit)

O_{ijklmn}	S_{ijklmn}	C_{ijklmn}	Mesin
O_{101031}	3290	3400	m_{31}
O_{111141}	3082.33	3120.33	m_{41}
O_{121121}	2915	3175	m_{21}
O_{122112}	2945	3290	m_{12}
O_{112132}	3085	3260	m_{31}
O_{201011}	3010	3250	m_{11}
O_{211131}	2373	2558	m_{31}
O_{221111}	2380.05	2855.05	m_{11}
O_{212121}	2380	2890	m_{21}
O_{222141}	2740	3010	m_{41}
O_{301021}	3190	3300	m_{21}
O_{311131}	2825	3065	m_{31}
O_{321113}	2250	3190	m_{13}

O_{ijklmn}	S_{ijklmn}	C_{ijklmn}	Mesin
O_{331232}	1624.15	2184.15	m_{31}
O_{341241}	1553.75	1828.75	m_{42}
O_{332241}	1630	2690	m_{41}
O_{342221}	1555	2335	m_{21}
O_{401031}	3715	3875	m_{31}
O_{412132}	3460	3715	m_{32}
O_{41141}	3451.67	3506.67	m_{41}
O_{422112}	3205	3715	m_{12}
O_{421132}	3060	3440	m_{32}
O_{501021}	3820	3950	m_{21}
O_{511141}	3535	3820	m_{41}
O_{521111}	3709	3820	m_{11}

Tabel 4.10 Waktu Mulai dan Selesai Operasi Penentuan *Due Date* (menit)

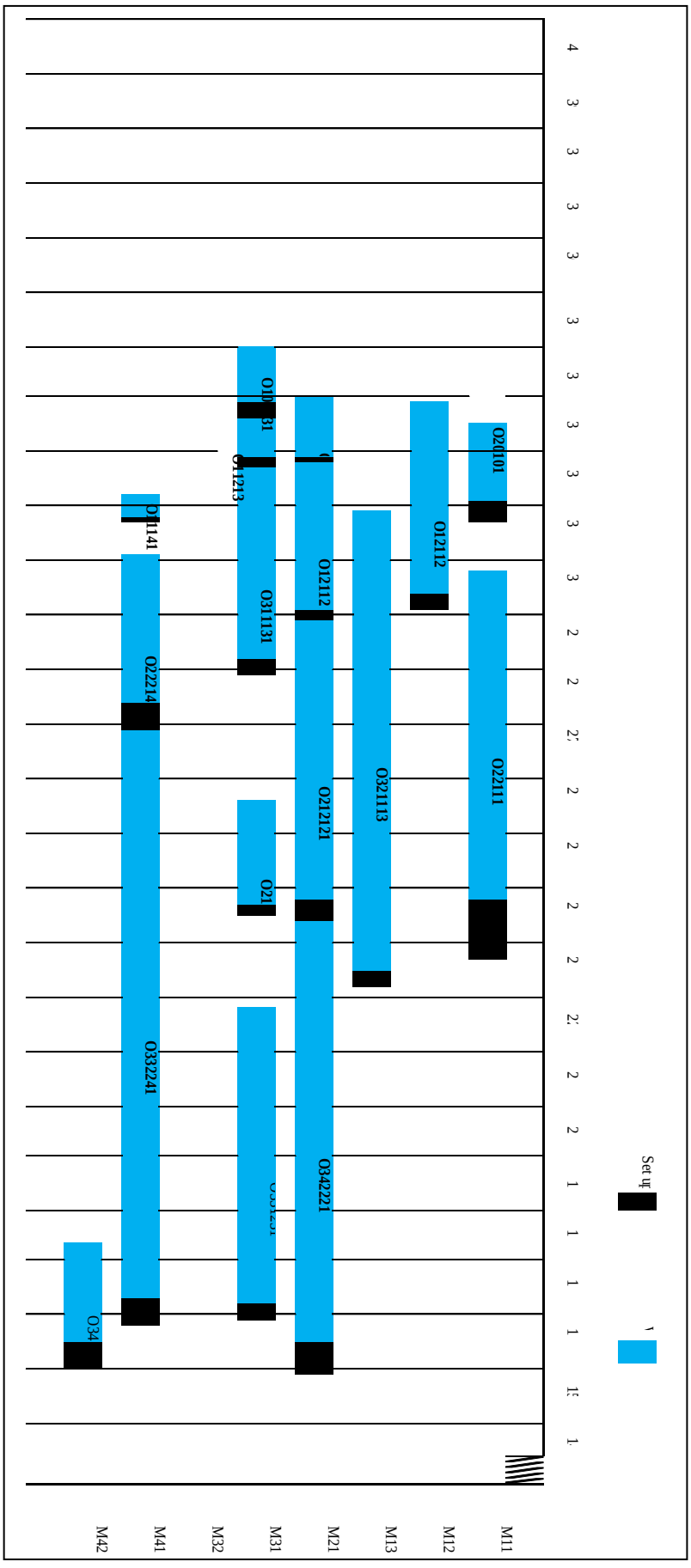
O_{ijklm}	S_{ijklmn}	C_{ijklmn}	Mesin
O_{10103}	390	500	m_{32}
O_{12112}	25	285	m_{21}
O_{12211}	45	390	m_{12}
O_{11114}	10	48	m_{41}
O_{11213}	20	195	m_{31}
O_{20101}	840	1080	m_{11}
O_{22111}	100	575	m_{11}
O_{21113}	210	395	m_{31}
O_{21212}	330	840	m_{21}

O_{ijklm}	S_{ijklmn}	C_{ijklmn}	Mesin
O_{22214}	459.95	729.95	m_{41}
O_{30102}	1920	2030	m_{21}
O_{31113}	1705	1945	m_{31}
O_{32111}	30	970	m_{13}
O_{33123}	430	990	m_{31}
O_{33224}	445	1505	m_{42}
O_{34124}	93	368	m_{41}
O_{34222}	900	1680	m_{21}

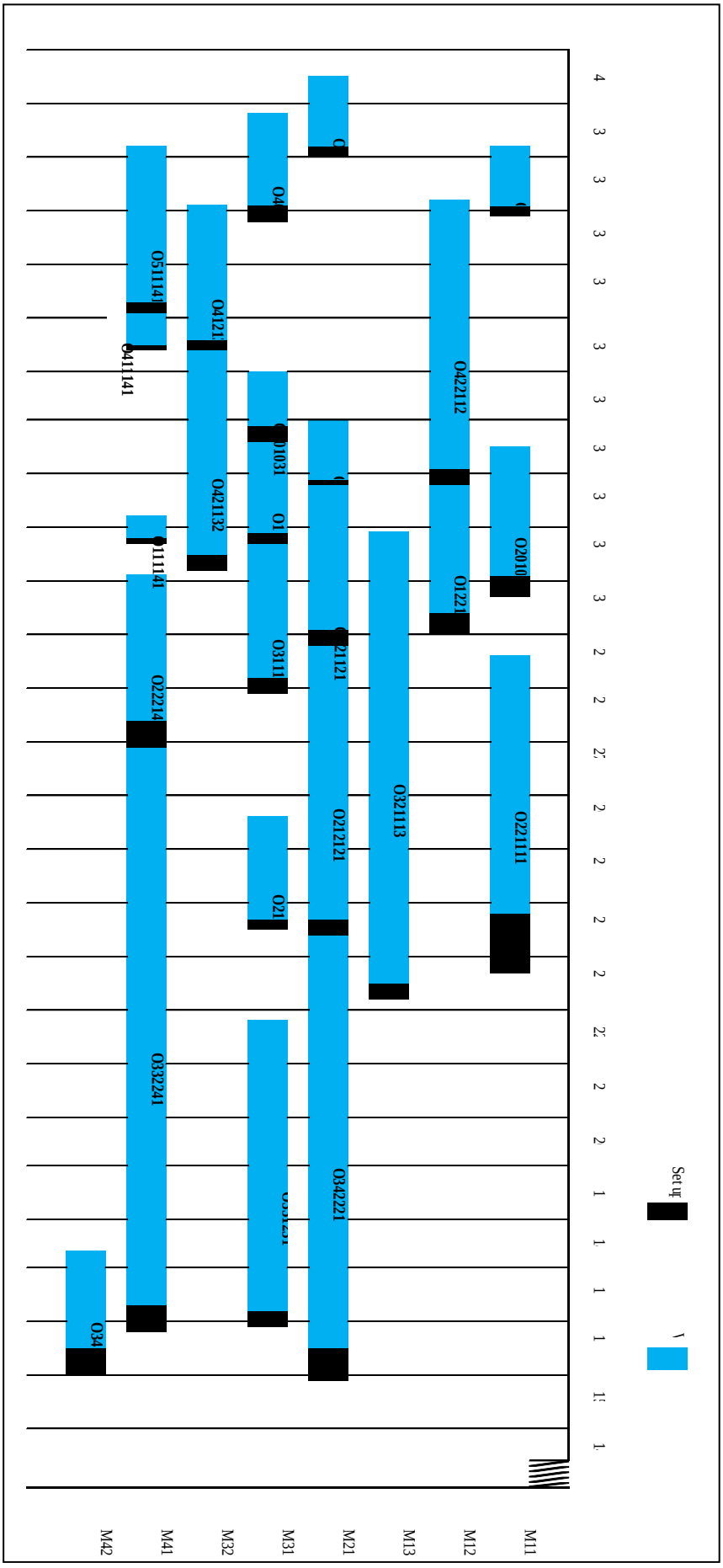
Sehingga waktu mulai dan selesai semua operasi dapat ditentukan due date dari setiap item, yang ditunjukan dari nilai waktu penyelesaian maksimum semua operasi pada suatu item. Dimana waktu penyelesaian maksimum tersebut, menunjukan waktu pekerjaan dapat diselesaikan dan siap untuk diserahkan kepada customer. *Due date* dari masing-masing item, sebagai berikut.

Tabel 4.11 *Due Date* Setiap Item

P_{i0}	10	20	30
Due Date (d_{i0})	500	1080	2030



Gambar 4.3 Gant Chart Akhir Pemenuhan *Due Date* Kondis Statis



Gambar 4.4 Gant Chart Akhir Pemenuhan *Due Date* Kondis Dinamis

Total actual flow time yang dihasilkan sebesar **3470** menit. Dimana item p_{10} baru dapat diselesaikan dan diserahkan kepada customer pada *t=500*, sedangkan untuk item p_{20} dan p_{30}, masing-masing pada t=1080 dan t=2030. Waktu-waktu penyelesaikan ini, sebagai acuan waktu penyelesaikan pekerjaan yang dapat dinegosiasikan kepada customer.

4.1.6 Hasil Pengujian Kondisi Dinamis Penentuan *Due- Date*

Dengan menggunakan algoritma penenuan *due date* pada model dua untuk kondisi dinamis, diperoleh waktu mulai dan selesai dari masing-masing operasi setelah terjadi kedatangan pesana baru pada waktu T=1100, seperti ditunjukan pada Tabel 4.12. Adapun *Gantt Chart* akhir untuk kondisi dinamis seperti ditunjukan pada Gambar 4.6, dengan detail perhitungan pada Lampiran 4.

Tabel 4.12 Waktu Mulai dan Selesai Penentuan Due Date Dinamis (menit)

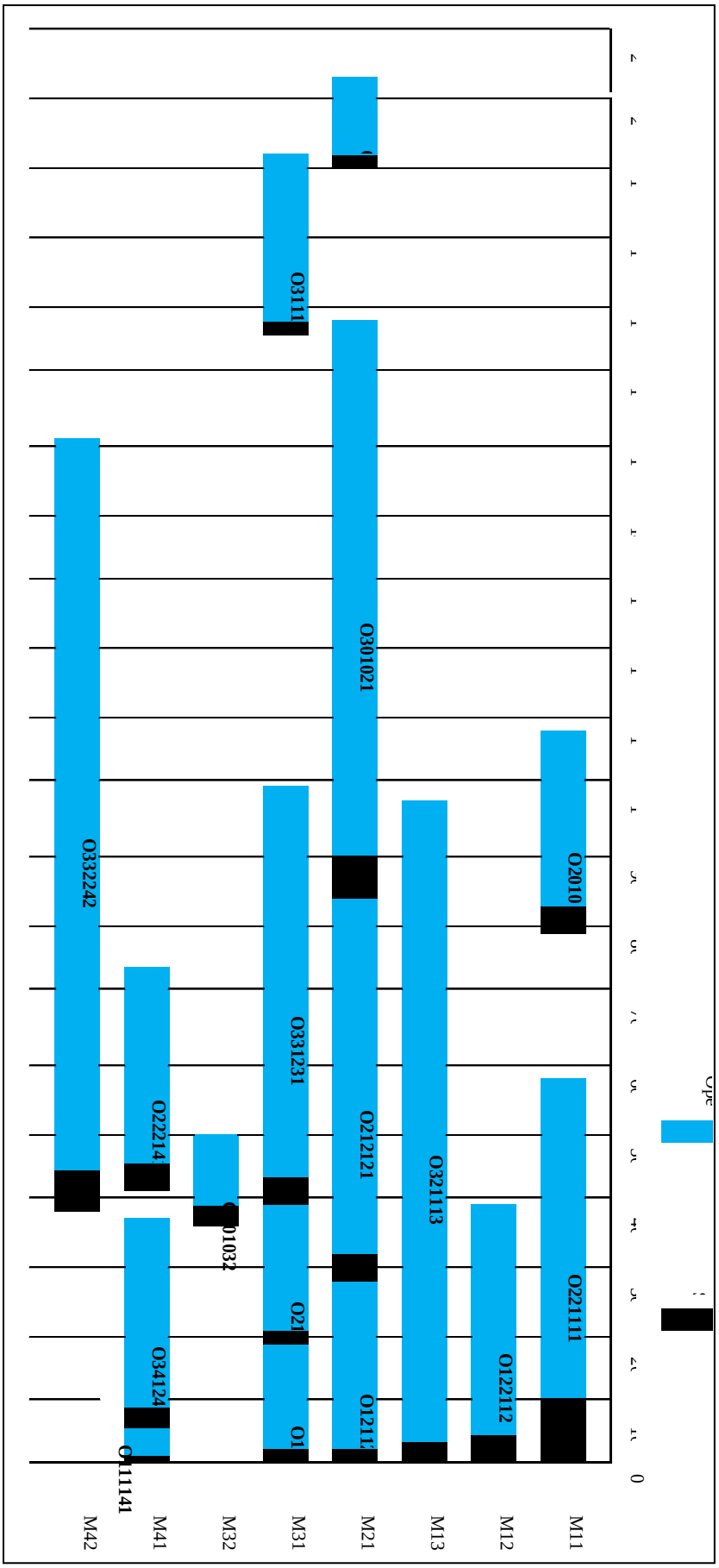
O_{ijklmn}	S_{ijklmn}	C_{ijklmn}	Mesin
O101032	390	500	M32
O121121	25	285	M21
O122112	45	390	M12
O111141	10	48	M41
O112131	20	195	M31
O201011	840	1080	M11
O221111	100	575	M11
O211131	210	395	M31
O212121	330	840	M21
O222141	459.95	729.95	M41
O301021	1920	2030	M21
O311131	1705	1945	M31
O321111	30	970	M13

O_{ijklmn}	S_{ijklmn}	C_{ijklmn}	Mesin
O331231	430	990	M31
O332242	445	1505	M42
O341241	93	368	M41
O342221	900	1680	M21
O401032	1705	1865	M32
O411141	1415	1470	M41
O412132	1425	1680	M32
O421131	1125	1505	M31
O422112	1145	1655	M12
O501021	1695	1825	M21
O511141	1120	1405	M41
O521111	1115	1226	M11

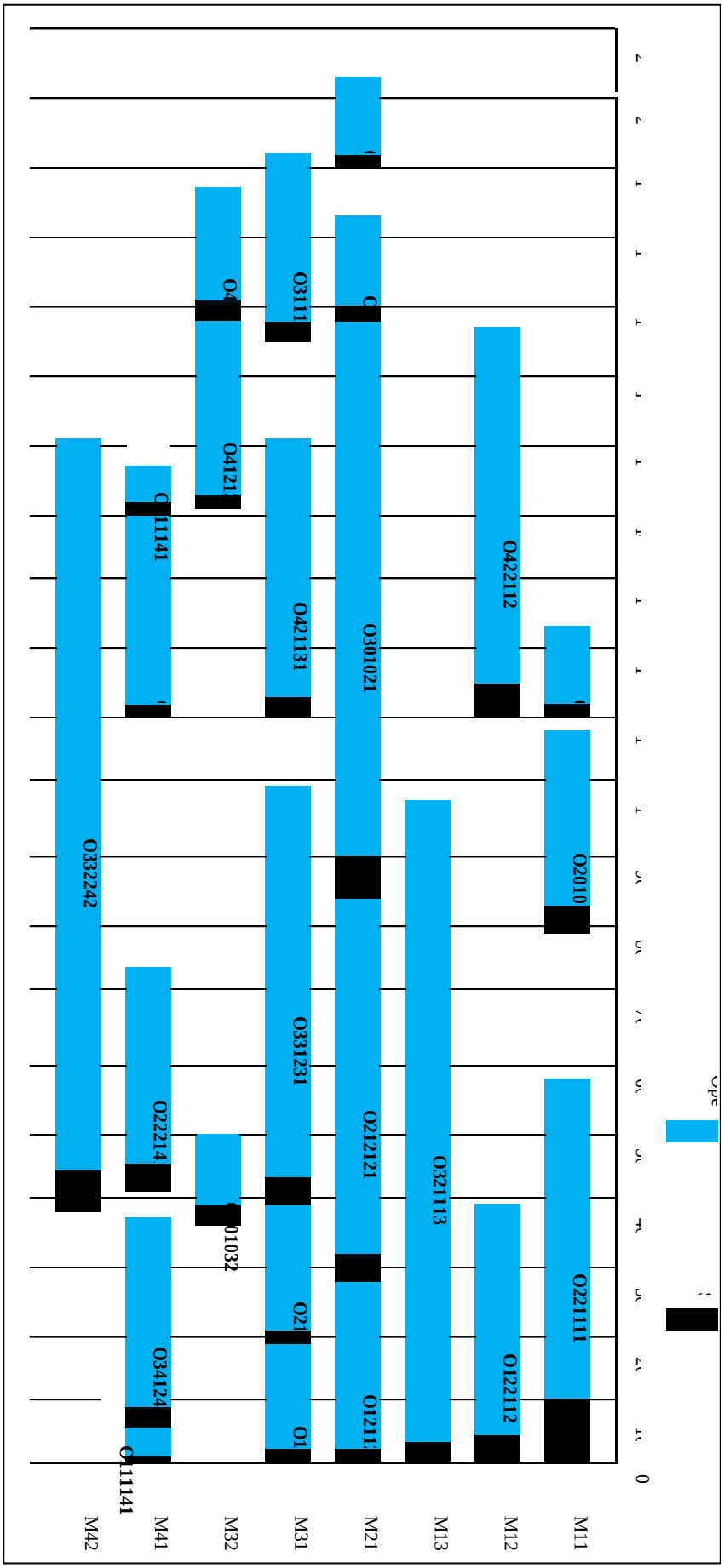
Setelah terjadi kedatangan pesanan baru, maka *TF* baru yang dihasilkan sebesar **4920**, dengan *due date* dari pesanan baru seperti ditunjukan pada Tabel 4.13, sebagai berikut.

Tabel 4.13 Due Date Item Pesanan Baru

P_{i0}	**40**	**50**
Due Date (d_{i0})	1865	1825



Gambar 4.5 Gant Chart Akhir Penentuan *Due Date* Kondis Statis



Gambar 4.6 Gant Chart Akhir Penentuan *Due Date* Kondis Dinamis

Berdasarkan dari hasil perhitungan yang dilakukan, terlihat bahwa *TF* pada kondisi dinamis didalam permasalahan penentuan *due date* lebih besar dibandingkan dengan *TF* didalam pemenuhan *due date*. Kondisi ini terjadi, dikarenakan perbedaan aturan prioritas penyelesaian pekerjaan pada kedua permasalahan tersebut.

4.2 Pengujian dan Analisa Model

Set data yang digunakan dalam pengujian model adalah data sistem manufaktur yang memproses sebanyak 10 jenis item dengan jumlah dan *due-date* yang berbeda-beda dan dikerjakan pada 5 buah mesin, dimana setiap mesin terdiri dari 3 buah mesin identik. Struktur produk seluruh item terdiri dari 4 *level* dan komponen pada suatu *level* tertentu memiliki sebanyak 2 buah komponen pembentuk pada *level* dibawahnya. Penetapan 2 buah komponen pembentuk ini (dengan jumlah unit masing masing komponen berbeda) dimaksudkan untuk dapat mewakili kondisi bahwa suatu komponen memiliki multi-komponen pembentuknya.

Tiap komponen memiliki sebanyak 2 buah operasi yang dikerjakan pada 2 buah mesin yang berbeda. Penetapan 2 buah operasi dimaksudkan untuk dapat mewakili kondisi bahwa suatu komponen memiliki multi-operasi, sementara pengerjaan pada 2 buah mesin yang berbeda adalah untuk menjaga tingkat variabilitas permasalahan penjadwalan. Tiap item dan komponen-komponennya juga memiliki waktu *set-up* dan waktu operasi (keduanya dalam satuan menit) pada tiap mesin dalam *routing* masing-masing yang besarnya berbeda beda. Serta *defect rate* yang berbeda untuk setiap tahapan pemprosesan atau mesin yang dilakui. Secara rinci data pengujian yang digunakan diperlihatkan pada Lampiran 5.

Pengujian dilakukan untuk mendapatkan jumlah alternatif jadwal untuk suatu jumlah jenis item tertentu dengan suatu jumlah *level* produk yang tertentu pula, serta rata-rata *CPU processing time* yang diambil dari 5 kali run program yang dilakukan. Proses pengujian dilakukan dengan program komputer yang dikembangkan dengan menggunakan bahasa *Java* dengan menggunakan *IDE Eclipse, version helios release* dan *build id:20100617-1415*. Dengan spesifikasi komputer yang digunakan *processor Intel® Core™2 Duo CPU T5870 @2.00 GHz, memory* 20148 GB dan sistem operasi *Window 7 Professional 32 Bit.*

4.2.1 Hasil Pengujian Pemenuhan *Due Date*

Dari hasil pengujian yang dilakukan diperoleh alternatif jadwal dan waktu rata-rata *CPU time* (dalam satuan detik), seperti ditunjukan pada Tabel 4.14 dan Tabel 4.15, sebagai berikut.

Tabel 4.14 Alternatif Jadwal Pemenuhan *Due Date*

	1 Level	**2 Level**	**3 Level**	**4 Level**
1 Item	15	44	123	365
2 Item	38	122	365	1.160
3 Item	60	218	749	2.449
4 Item	88	351	1.279	4.298
5 Item	121	518	1.944	6.615
6 Item	154	713	2.599	9.311
7 Item	183	914	3.451	11.791
8 Item	222	1.145	4.301	15.406
9 Item	277	1.426	5.535	19.491
10 item	326	1.706	6.306	22.886

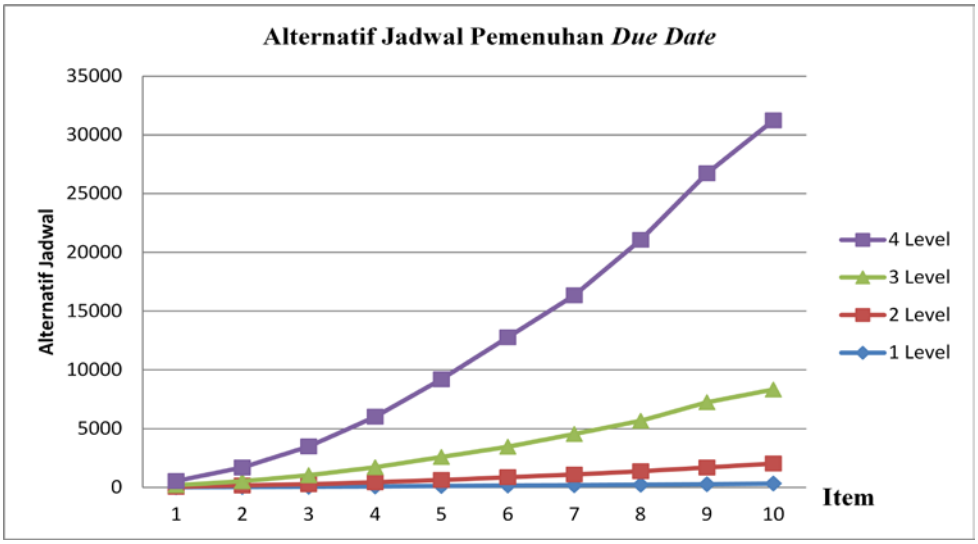
Tabel 4.15 CPU Time Pemenuhan *Due Date* (dalam detik)

	1 Level	2 Level	3 Level	4 Level
1 Item	0.025	0.053	0.090	0.162
2 Item	0.059	0.090	0.147	0.415
3 Item	0.075	0.125	0.253	0.902
4 Item	0.089	0.173	0.392	1.603
5 Item	0.090	0.184	0.555	2.962
6 Item	0.096	0.215	0.918	4.694
7 Item	0.112	0.278	1.270	7.929
8 Item	0.113	0.434	1.405	12.200
9 Item	0.120	0.343	1.864	13.969
10 item	0.134	0.431	2.415	19.336

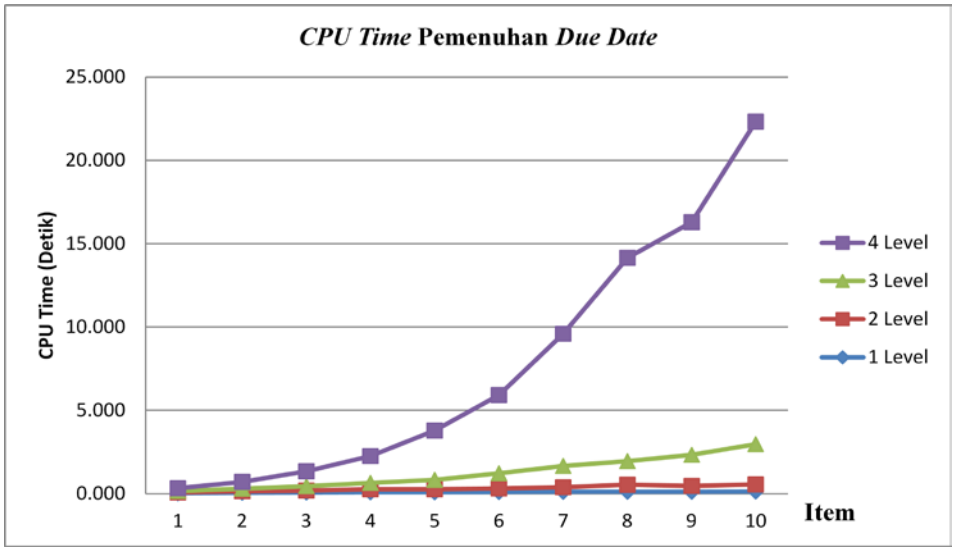
Tabel 4.14 menunjukan jumlah alternatif jadwal yang dapat terbentuk dari setiap kondisi perlakukan, untuk kondisi 1 item sampai dengan 10 item dengan level yang digunakan mulai dari 1 level sampai dengan 4 level. Sedangkan Tabel 4.15 menunjukan waktu *CPU time* yang diperlukan untuk pencarian solusi dari setiap permasalahan sesuai kondisi perlakukan yang diberikan.

Dari kedua tabel tersebut diatas terlihat bahwa algoritma yang dikembangkan didalam penelitian ini, dapat menyelesaikan permasalahan dengan memberikan sejumlah alternatid jadwal yang dapat terbentuk, sehingga dikatakan

memiliki tingkat keberlakuan yang umum. Adapun grafik hubungan antara jumlah item dengan level terhadap peningkatan alternatif jadwal dan *CPU time*, seperti ditunjukan pada Gambar 4.7.



Gambar 4.7 Grafik Alternatif Jadwal Pemenuhan Due Date



Gambar 4.8 Grafik *CPU Time* Pemenuhan Due Date

4.2.2 Hasil Pengujian Penentuan *Due Date*

Dari hasil pengujian yang dilakukan diperoleh alternatif jadwal dan waktu rata-rata *CPU time* (dalam satuan detik), seperti ditunjukan pada dan Tabel 4.15, sebagai berikut.

Tabel 4.16 Alternatif Jadwal Penentuan *Due Date*

	1 Level	2 Level	3 Level	4 Level
1 Item	17	46	124	356
2 Item	40	115	344	1.044
3 Item	65	196	600	2.216
4 Item	95	288	908	3.632
5 Item	128	463	1.553	5.607
6 Item	166	644	2.047	7.548
7 Item	202	874	2.668	9.384
8 Item	238	1.024	3.394	12.068
9 Item	280	1.312	4.223	18.052
10 item	334	1.466	4.774	21.657

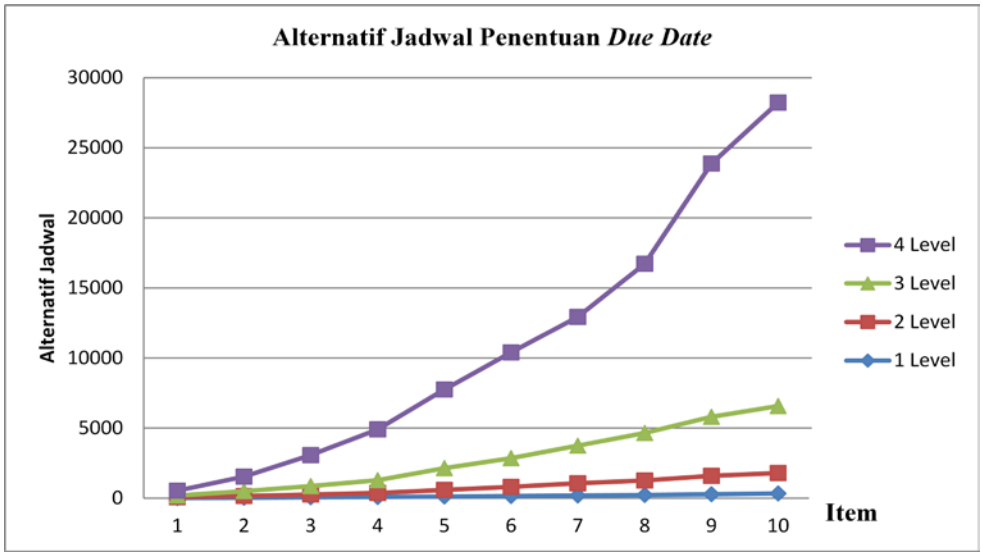
Tabel 4.17 CPU Time Penentuan *Due Date* (dalam detik)

	1 Level	2 Level	3 Level	4 Level
1 Item	0.034	0.053	0.128	0.227
2 Item	0.050	0.085	0.247	1.466
3 Item	0.062	0.159	0.618	5.351
4 Item	0.075	0.184	1.361	15.921
5 Item	0.087	0.303	2.679	31.011
6 Item	0.081	0.415	4.777	59.787
7 Item	0.103	0.627	7.960	102.635
8 Item	0.116	0.928	11.686	165.981
9 Item	0.172	1.271	21.461	252.036
10 item	0.193	1.771	26.642	381.426

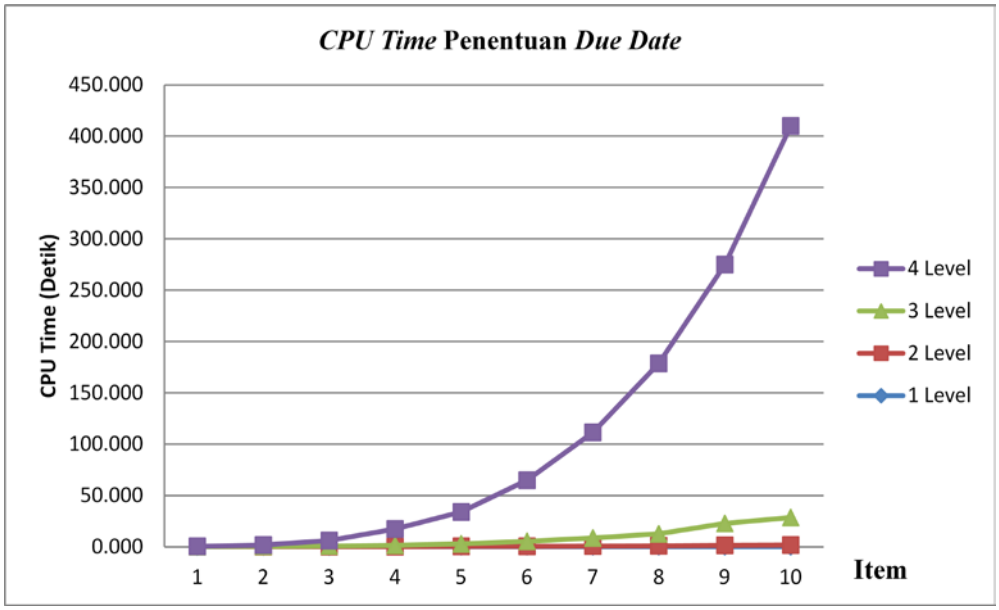
Tabel 4.16 menunjukan jumlah alternatif jadwal yang dapat terbentuk dari setiap kondisi perlakukan penentuan *due date* dari masing-masing item, untuk kondisi 1 item sampai dengan 10 item dengan level yang digunakan mulai dari 1level sampai dengan 4 level. Sedangkan Tabel 4.17 menunjukan waktu *CPU time* yang diperlukan untuk pencarian solusi dari setiap permasalahan sesuai kondisi perlakukan yang diberikan.

Dari kedua tabel tersebut diatas terlihat bahwa algoritma yang dikembangkan didalam penelitian ini, dapat menyelesaikan permasalah dengan memberikan sejumlah alternatid jadwal yang dapat terbentuk . Sehingga dikatakan memiliki tingkat keberlakuan yang umum, didalam penyelesaian permasalahan penentuan *due date* pada lingkungan produksi *job shop* dinamis mesin paralel yang memproduksi multi item berstruktur multi level. Adapun grafik hubungan

antara jumlah item dengan level terhadap peningkatan alternatif jadwal dan *CPU time*, seperti ditunjukan pada Gambar 4.9 dan Gambar 4.10.



Gambar 4.9 Grafik Alternatif Jadwal Penentuan *Due Date*



Gambar 4.10 Grafik *CPU Time* Jadwal Penentuan *Due Date*

4.2.3 Analisa Model

Berdasarkan pada data-data yang diperoleh pada sub bab 4.2.1 dan sub bab 4.2.2, dapat dilakukan analisa untuk hal-hal sebagai berikut.

4.2.3.1 Peningkatan Jumlah Item

Peningkatan jumlah item pada tingkatan level tertentu memberikan pengaruh terhadap peningkatan alternatif jadwal yang dapat terbentuk secara

signifikan. Misalkan pada kondisi 2 level dengan jumlah item sebanyak 3 buah jumlah altenatif jadwal sebanyak 60 alternatif, sedangkan derngan jumlah item sebanyak 4 buah terbentuk sebanyak 88 alternatif jadwal pada permasalahan pemenuhan *due date* artinya terjadi hampir 1,46 kali peningkatan alternatif jadwal. Sedangkan pada permasalahan penentuan *due date* yang sama terbentuk sebanyak 65 alternatif jadwal dan 95 alternatif jadwal. Untuk 4 level dengan 3 buah item dihasilkan pada permasalahan pemenuhan *due date* sebanyak 2.449 alternati dan pada kondisi dengan 4 buah item sebanyak 4.298 alternatif peningkatan yang terjadi sebesar 1,75 kali lipat.

Pada kondisi penentuan *due date* dengan perlakuan 3 level, dengan item yang diproses sebanyak 3 dan 4 buah item, alternatif jadwal yang terbentuk sebanyak 2.216 dan 3.632 alternatif jadwal, terjadi peningkatan sebesar 1,51 kali. Peningkatan alternatif yang jadwal yang terbentuk berpengaruh terhadap waktu *CPU time* yang diperlukan untuk melakukan penyusunan operasi-operasi sesuai dengan urutan pemprosesannya. Pada kondi 3 item dengan 2 level pada permasalahan pemenuhan dan penentuan *due date* masing-masing waktu diperlukan sebesar 0.125 dan 0.159 detik, sedangkan dengan 4 level diperlukan waktu sebesar 0.173 dan 0.184 detik. Sehingga peningkatan waktu sebesar 1,38 kali dan 1,15 kali.

Kondisi di atas menunjukkan bahwa peningkatan jumlah jenis item akan memberikan peningkatan jumlah alternatif jadwal yang besarnya berbeda untuk 2 dan 3 buah *level* produk, bahwa pada jumlah *level* produk yang lebih banyak akan memberikan peningkatan yang lebih besar, yang akan berdampak pada *CPU time* yang diperlukan untuk mendapatkan solusi yang dapat memberikan *total actual flow time* yang minimum.

4.2.3.2 Peningkatan Jumlah Level

Peningkatan jumlah level yang diberikan memberikan pengaruh yang signifikan terhadap alternatif jadwal yang terbentuk. Misalkan pada kondisi 2 item dengan 2 level pada kondisi pemenuhan dan penentuan *due date* masing-masing memberikan alternatif jadwal sebanyak 122 dan 115 alternatif jadwal. Sedangkan dengan kondisi sebanyak 3 level diperoleh sebanyak 365 dan 344

alternatif jadwal. Sehingga terjadi peningkatan alternatif jadwal sebesar 2.99 kali dan 2.99 kali.

Adapun waktu *CPU time* yang diperlukan untuk pencarian solusi dari permasalahan 2 item dengan 2 level , untuk permasalahan dan penentuan *due date* masing-masing terjadi sebesar 0.090 detik dan 0.084 detik, sedangkan untuk 2 item dengan 4 level sebesar 0.415 detik dan 1.466 detik. Sehingga terjadi peningkatan waktu pencarian solusi sebesar 4.6 kali 17.4 kali.

Kondisi di atas menunjukkan besarnya peningkatan jumlah alternatif jadwal dengan terjadinya peningkatan jumlah *level* produk adalah berbeda pada 2 buah jenis item, bahwa peningkatannya akan lebih besar untuk jumlah level dengan lebih banyak lagi. Selain itu terlihat juga bahwa peningkatan waktu pencarian solusi pada permasalahan penentuan *due date* meningkat jauh lebih besar dibandingkan dengan permasalahan pemenuhan *due date*.

4.2.3.3 Peningkatan Jumlah Item dan Level

Peningkatan rata-rata jumlah alternatif jadwal dengan terjadinya peningkatan jumlah jenis item adalah 1.57 kali dan sebesar 3.75 kali untuk peningkatan jumlah *level* produk, pada permasalahan pemenuhan *due date*. Sedangkan ada permasalahan penentuan *due date* terjadi rata-rata peningkatan alternatif jadwal dengan peningkatan jumlah item sebesar 1.54 kali dan sebesar 3.49 kali dengan peningkatan level.

Dari segi *CPU time* Peningkatan rata-rata jumlah alternatif jadwal dengan terjadinya peningkatan jumlah jenis item adalah 1.42 kali dan sebesar 3.60 kali untuk peningkatan jumlah *level* produk, pada permasalahan pemenuhan *due date*. Sedangkan ada permasalahan penentuan *due date* terjadi rata-rata peningkatan alternatif jadwal dengan peningkatan jumlah item sebesar 1.78 kali dan sebesar 8.23 kali dengan peningkatan level. Dari kedua besaran peningkatan rata-rata ini dapat dilihat bahwa peningkatan jumlah *level* item memberikan angka yang lebih tinggi daripada peningkatan jumlah jenis item.

Dari perbandingan tersebut, maka dapat dikatakan bahwa peningkatan yang terjadi pada jumlah *level* item akan memberikan peningkatan jumlah alternatif jadwal yang lebih signifikan daripada peningkatan pada jumlah jenis

item. Sehingga peningkatan jumlah *level* produk lebih mempengaruhi tingkat kompleksitas permasalahan penjadwalan dan peningkatan pada jumlah alternatif jadwal bersifat bersifat lebih sensitif terhadap peningkatan jumlah level produk dibandingkan peningkatan jumlah jenis item, yang berdampak pada peningkatan *CPU time*.

4.2.3.4 Pemenuhan dan Penentuan *Due Date*

Didalam permasalahan pemenuhan dan penentuan *due date* terjadi adanya peningkatan alternatif jadwal yang dipengaruh oleh jumlah item dan jumlah level dari setiap item. Pada suatu kondisi alternatif yang terbentuk pada pemenuhan jumlah *due date* lebih sedikit tetapi pada kondisi yang berbeda alternatif jadwal pada permasalah penentuan *due date* lebih sedikit dibandingkan dengan pemenuhan *due date*. Misalkan pada kondisi 1 level dengan 10 item alternatif jadwal yang pada pemenuhan due date lebih sedikit dibandingkan dengan penentuan *due date*, tetapi untuk level lebih dari 1 alternatif jadwal yang terbentuk pada permasalahan penentuan due date relatif lebih sedikit di bandingkan dengan pemenuhan *due date*.

Dari segi *CPU time* pada permasalah penentuan *due date* relatif lebih tinggi dibandingkan dengan pemenuhan *due date*, dapat terlihat mulai dari kondisi 1 level dengan 7 item dan 2 level dengan 3 item. *CPU time* yang lebih tinggi, meskipun jumlah alternatif jadwal yang terbentuk lebih sedikit dibandingkan dengan permasalahan pemenuhan *due date*. Kondisi diakibatkan dalam permasalahan penentuan *due date* pada saat memulai proses dapat dilakukan secara random selama proses yang bersangkutan tidak memiliki proses pendahulu pada saat awal penjadwalan, sehingga perlu dilakukan pergeseran jadwal hingga didapatkan jadwal memberikan *total actual flow time* yang minimum. Sedangkan pada permasalahan pemenuhan *due date* proses yang dijadwalkan pertama sudah ditentukan yaitu proses paling akhir untuk level 0 atau pada tingkat item.

BAB V
SARAN DAN KESIMPULAN

5.1 Kesimpulan

Kesimpulan yang dapat diambil dari penelitian tugas akhir ini adalah sebagai berikut:

1. Algoritma yang dikembangkan terbukti dapat menyelesaikan permasalahan pemenuhan dan penentuan *due date* baik pada kondisi statis dan dinamis, dengan tujuan meminimal *total actual flow time* dan mempertimbangkan *defect rate*.
2. Algoritma yang disusun memiliki tingkat keberlakuan yang umum dengan dihasilkannya jumlah alternatif jadwal berdasarkan pengujian yang dilakukan dari kombinasi 10 item dan 4 level.
3. Didapatkan pula bahwa peningkatan jumlah item dan jumlah level pada suatu jumlah jenis item memberikan peningkatan yang signifikan pada jumlah alternatif jadwal dan waktu yang diperlukan untuk menyelesaikan penyusunan jadwal. Peningkatan jumlah level produk pada jumlah jenis item yang sama, memberikan peningkatan dengan besaran yang berbeda pada jumlah alternatif jadwal. Peningkatan pada jumlah jenis item yang lebih sedikit memberikan peningkatan jumlah alternatif jadwal yang lebih kecil daripada jumlah jenis item yang lebih banyak.
4. Peningkatan pada jumlah level produk memberikan peningkatan yang lebih besar daripada peningkatan pada jumlah jenis item terhadap jumlah alternatif jadwal dan tentunya juga terhadap waktu yang diperlukan untuk penyelesaian proses pengurutan operasi. Perbandingan tersebut menunjukkan bahwa tingkat kompleksitas permasalahan penjadwalan lebih dipengaruhi oleh peningkatan jumlah level produk dan peningkatan pada jumlah alternatif jadwal lebih sensitif terhadap peningkatan jumlah level produk daripada peningkatan jumlah jenis item.
5. Alternatif jadwal yang dapat terbentuk untuk kondisi penentuan *due date* relatif lebih banyak di bandingkan dengan pemenuhan *due date* pada kondisi lebih dari 1 level dan lebih dari 2 item, sedangkan untuk kondisi lainnya

berlaku sebalinya. Tetapi *CPU time* yang diperlukan lebih besar didalam penyelesaian permasalahan pemenuhan *due date.*

5.2 Saran

Dari hasil penelitian tugas akhir ini, dapat dilakukan pengembangan seperti berikut ini.

1. Pengembangan model yang memungkinkan dilakukannya proses penentuan jumlah dan ukuran *batch* (*batching*), pengurutan operasi item, dan penjadwalan *batch* yang dihasilkan secara simultan.
2. Penelitian untuk menghasilkan algoritma penjadwalan pada kondisi mesin parallel dengan mempertimbangkan aktivitas perawatan atau *breakdown machine*.
3. Pengembangan model yang memperhatikan aspek *financial* dalam mengambil keputusan terhadap kedatangan pesanan baru berdasarkan keuntungan dan penalti dari konsumen atas keterlambatan penyelesaian pekerjaan.
4. Pengembangan model penentuan *due date* dengan ukuran *batch integer*, karena banyak kasus *batch integer* lebih mendekati kondisi nyata..
5. Pengembangan algoritma-algoritma yang lebih efisien didalam penyelesaian permasalahan dari penelitian tugas akhir ini.

DAFTAR PUSTAKA

Baker,K.R (1974). *Introduction to Sequencing and scheduling,* New York, John Willey & Sons, Inc.

Baker,K.R andTrietsch (2009). *Introduction to Sequencing and scheduling,* New York, John Willey & Sons, Inc.

Dobson, G., Karmarkar, U.S., and Rummel, J.L (1987). *Batching to Minimize Flow Times on One Machines,* Management Science, Vol. 33.pp.784-799.

Halim, A.H., Ohta, H. (1993). Batch Scheduling Problems of Multiple Items Through the Flow Shop with both Receiving and Delivery Just in Times, *International Journal of Operation Research,* 31,pp. 1943-1955.

Bedworth, D.D., and Bailey, J.E (1987). *Integrated Production Control Systems,* New York, John Willey & Sons, inc.

Brucker, Peter., dan Knust, Sigrid. (2007). *Complex Scheduling*, Springer, Germany.

French, Simon. (1982). *Squencing and Scheduling: An Introduction to the Mathematics of the Job Shop,* Chichester, Ellis Horwood.

Sotskov, Y.N., Tautenhahn, Y., and Werner, F. On The Application of Insertion Techniques for Job Shop Problems with Setup Times, *RAIRO Operations Research,* Vol. 33, No. 2,1999,pp. 209-245.

Artigues et al.(2003). Insertion Technique for Static and Dynamic Resource Constrained Project Scheduling. *European Journal of Operation Research,*149, pp.249-267.

Vestjens, Arjen P,A et all (2007). Complexity of the Job Insertion Problem in Multi Stage scheduling. *Operation Research Letter,* 35.pp 754-758.

Mosheiov, Gur., Oron, Daniel. (2006). Due-date Assignment and Maintenance Activity Scheduling Problem, *Mathematical and Computer Modeling Journal*,Vol.44, pp 1053-1057.

Huang, R.H.,Yang,C.L. (2010) Multi-objective job-shop scheduling with lot-splitting production. *International Journal of Production Economics*.,pp.206-213

Khasnan, Ali Husseinzadeh et al,.(2008). A Hybrid Genetic herutistic for Scheduling Paralel Batch Processing Machines With Arbitrary Job Sizes. *Journal Computer and Operation Research.* Vol 35, pp.1084-1098.

Cheng, TCE et al., (2007).Due-Date Assignment And Paralle Machine Scheduling With Deteriorating Jobs. *Journal of Operational Research Society*, Vol.58, No.8, pp.1103-1108.

Xia,Yu et al.(2008), Job Squencing and Due Date Assignment in a Single Machine Shop with Uncertain Processing Times. *European Journal of Operational Research*. Vol.184. pp.63-75.

Shabtay,Dvir. (2010).Scheduling and Due Date Assignment to Minimize Earliness, Tardiness, Holding, Due Date Assignment and Batch Delibery Costs. *International Journal of Production Economics*. Vol 123, pp.235-242.

Tuong, Nguyen Huynh and Soukhal Ameur. (2010). Due Date Assignment and JIT Scheduling with Equal-Size Jobs. *European Journal of Operational Research.*Vol 205, pp.280-289.

Biskup, Dirk., Jahnke, Hermann. (2001),"Common Due-date Assignment for Scheduling om a Single Machine with Jointly Reducible Processing Times," *International Journal of Production Economics*, pp.317-322, Elsevier.

Shabtay, Dvir., Steiner, George., (2005),"*Two Due-Date Assignment Problem in Scheduling a Single Machine*",*Operation Research Letters*.

Vollmann, Thomas E., et al,(2005). *Manufacturing Planning and Control for Supply Chain Management*,pp.300-330,503-530,539-565. McGrawHill, New York.

Vinod, V., R. Sriharan. (2011). Simulation Modeling and Analysis of due-date Assignment Methods and Scheduling Decision Rules in Dynamic Job Shop Production System. *International Journal Production Economics*, Vol.129,pp.127-146.

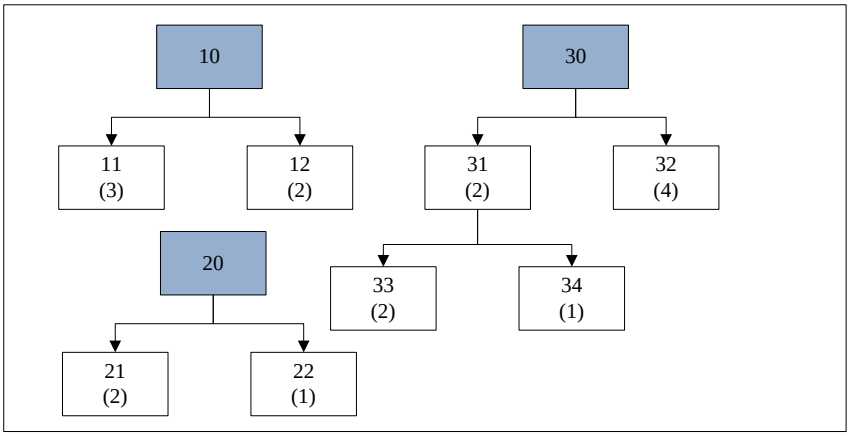
Chan, F.T.S et al. (2008). Lot Streaming for Product Assembly in Job Shop Environment. *Robotics and Computer-Integrated Manufacturing*, Vol.24. pp.321-331.

Thiagarajan, S., Candrasekharan Rajendran,(2005). Scheduling in Dynamic Assembely Job-Shops to Minimize the Sum of Weighted Earliness, Weighted Tardiness and Weighted Flow Time of Jobs. *Computer and Industrial Engineering*, Vol.49. pp.463-503.

Wong, T.C, et al (2009). A Resource-Constrained Assembly Job Shop Scheduling Problem with Lot Streaming Technique. *Computer and Industrial Engineering*, Vol.57, pp.983-995.

Pathumnakul, Supachai., Pius J. Egbelu, (2006). An Algorithm for Minimizing Weighted Earliness Penalty in Assembly Job Shops. *International Journal of Production Economics*,Vol.103, pp.230-245.

Perhitungan Pemenuhan Due Date Kondisi Statis



Gambar 1.1 Struktur Setiap Produk

Tabel 1.1 Data Mesin Tersedia

No	Mesin	Jumlah (Unit)
1	1	3
2	2	1
3	3	2
4	4	2

Tabel 1.2. Data Item, Kuantitas dan *Due Date*

Produk	Kuatitas (unit)	*Due-date* (menit)*
10	10	3400
20	15	3250
30	20	3300

Tabel 1.3 Waktu Proses, Set Up dan Tingkat Cacat

Item	Operation	Mesin	*Set-up time* (menit)	Waktu proses (menit)	Tingkat Cacat (%)
10	1	3	30	10	4
20	1	1	40	15	5
30	1	2	15	5	8
11	1	4	10	1	6
	2	3	20	5	5
12	1	2	25	10	10
	2	1	45	15	4
21	1	3	15	5	8
	2	2	45	15	6
22	1	1	100	25	5
	2	4	50	15	7

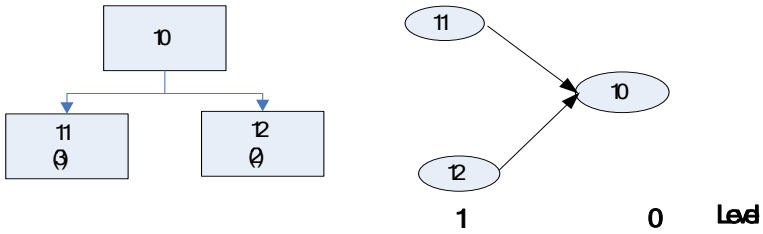
Item	Operation	Mesin	*Set-up time* (menit)	Waktu proses (menit)	Tingkat Cacat (%)
31	1	3	25	5	8
32	1	1	30	10	6
33	1	3	35	5	5
	2	4	50	10	10
34	1	4	45	5	5
	2	2	60	15	8

Dengan menggunakan algoritma kondisi statis, maka diperoleh langkah-langkah penyelesaian sebagai berikut

Langkah 0 : $P(P_{i0}) = \{ P_{10},P_{20},P_{30}\}$

Langkah 1 : $P'(P_{i0}) = \{ P_{10}, P_{30},P_{20} \}$

Langkah 2 : $P_{i0} = P_{10}$ dengan α=1



Langkah 3 :

Langkah 4 : P_{ij}={P_{11},P_{12}}, dengan m = {(m_4,m_3),(m_2,m_1)}

Langkah 5 :

P_{i0}	n_{i0}	dr_{i0} (%)	$n_{i0\ (diproduksi)}$	$O_{i0klm\bullet}$	$S_{ioklm\bullet}$	$t_{i0klm\bullet}$	d_{i0}
P_{10}	10	4	11	O_{10103}	30	10	3400

Langkah 6 :

P_{ij}	$Z(P_{ij})$	H_{ij}
P_{11}	P_{10}	3
P_{12}	P_{10}	2

Langkah 7 :

P_{ij}	n_{ij}	dr_{ijkm}(%)	$n_{ij(diproduksi)}$	O_{ijkm}	$s_{ijkm\bullet}$	$t_{ijkm\bullet}$
P_{11}	35	6	38	O_{11114}	10	1
	33	5	35	O_{11213}	20	5
P_{12}	23	10	26	O_{12112}	25	10

	22	4	23	O_{12211}	45	15

Langkah 8 : *MiLF* level 0 (l=0).

Langkah 9 : Gunakan Sub Algoritma Penentuan jumlah dan ukuran batch.

Langkah 1: $L(P_{ij})$ ={ P_{10}} dengan indeks x=1

Langkah 2: $M(P_{10})$ = {m_3}

Langkah 3: $N_{1,103}$=1 dengan $Q_{1,103[1]}$ = 11

Langkah 4: *Pilih* x=1

Langkah 5: Penjadwalan sesuai *routing* menghasilkan $F^a_{N=1} = 140$ Karena hanya diproses pada satu mesin,

Langkah 14: set x=x+1= 1+1=3, x>y, maka didapat solusi optimal.

P_{i0}	N_1	$Q_{10113[N]}$
P_{10}	1	11

Langkah 10 : set l=l+1= 0+1=1, kembali kelangkah 9.

Langkah 9 : Gunakan Sub Algoritma ukuran batch dan jumlah batch.

Langkah 1 : $L(P_{ij})$ ={ P_{11}, P_{12}} dengan indeks x=1 dan x=2.

Langkah 2 : $M(P_{11})$ = {m_4, m_3} dan $M(P_{12})$ = {m_2, m_1}

Langkah 3 : $N_{1,114}$=1 dan $Q_{1,114}$=38

$N_{1,213}$=1 dan $Q_{1,213}$=35

$N_{2,112}$=1 dan $Q_{2,112}$=26

$N_{2,121}$=1 dan $Q_{2,211}$=23

Langkah 4 : Pilih komponen dengan indeks x=1

Langkah 5 : Penjadwalan sesuai *routing* menghasilkan

$F^a_{N=1} = 223$

Langkah 6 : $M(P_{11})$={m_4} dengan s_4=10 dan t_4= 1

Langkah 7 : Set $N_{1,114}$=2

Langkah 8 : Diperoleh ukuran *batch* $Q_{1,14[1]}$= 24 dan $Q_{1,14[2]}$=14.

Langkah 9 : $Q_{x,km[u]}>0$, Ya.

Langkah 10: Tidak, berada pada operasi pertama

Langkah 11: Penjadwalan sesuai routing menghasilkan $F^a_{N=2} = 209$

(209≤223)

Langkah 12: Set $N_{1,114}$=3

Langkah 8 : Diperoleh ukuran *batch* $Q_{1,114[1]}$= 22,67, $Q_{1,114[2]}$=12,67, dan $Q_{1,114[3]}$= 2,66.

Langkah 9 : $Q_{x,km[u]}>0$, Ya.

Langkah 10: Tidak, berada pada operasi pertama

Langkah 11: Penjadwalan sesuai routing menghasilkan $F_{N=3}^{a} = 207.67$ (207.67≤209)

Langkah 12: Set $N_{1,114}$=4, dan kembali ke langkah 8

Langkah 8 : Diperoleh ukuran *batch* $Q_{1,114[1]}$= 24,5, $Q_{1,114[2]}$=14,5, $Q_{1,114[3]}$= 4,5 dan $Q_{1,114[4]}$= -5.5.

Langkah 9 : $Q_{x,klm[u]}>0$, Tidak terpenuhi, Set $N_{1,114[optimal]}$ =3 lanjut ke langkah 13

Langkah 13: Set o={m_3} dengan s_3=20 dan t_3= 5, kembali ke langkah 7.

Langkah 7 : Set $N_{1,213}$=2.

Langkah 8 : Diperoleh ukuran *batch* $Q_{1,213[1]}$= 19,5 dan $Q_{1,213[2]}$=15,5.

Langkah 9 : $Q_{x,klm[u]}>0$, ya.

Langkah 10: Ya, berada pada operasi terakhir

Langkah 12: Set $N_{1,213}$=3, dan kembali ke langkah 8.

Langkah 8 : Diperoleh ukuran *batch* $Q_{1,213[1]}$= 15,67, $Q_{1,213[2]}$=11,67 dan $Q_{1,213[3]}$=7,66.

Langkah 9 : $Q_{x,klm[u]}>0$, ya.

Langkah10 : Ya, berada pada operasi terakhir.

Langkah 12: Set $N_{1,213}$=4.dan kembali ke langkah 8.

Langkah 8 : Diperoleh ukuran *batch* $Q_{1,213[1]}$= 14,75, $Q_{1,213[2]}$=10,75, $Q_{1,213[3]}$=6,75 dan $Q_{1,213[4]}$=2,75.

Langkah 9 : $Q_{x,klm[u]}>0$, ya.

Langkah10 : Ya, berada pada operasi terakhir.

Langkah 12: Set $N_{1,213}$=5, dan kembali ke langkah 8.

Langkah 8 : Diperoleh ukuran *batch* $Q_{1,23[1]}$= 15, $Q_{1,23[2]}$=11, $Q_{1,23[3]}$=7, $Q_{1,23[4]}$=3 dan $Q_{1,23[5]}$=-1.

Langkah 9 : $Q_{x,klm[u]}>0$, Tidak. set $N_{1,23[optimal]}$ = 4, lanjut kelangkah 13.

Langkah 10 : Tidak, berada pada operasi pertama

Langkah 11 : Penjadwalan sesuai routing menghasilkan $F_{N=4}^{a} =$ 472(472≤472.5)

Langkah 12: Set $N_{2,12}$= 6, kembali ke langkah 8.

Langkah 8 :Diperoleh ukuran *batch* $Q_{2,112[1]}$= 10,58, $Q_{2,112[2]}$=8,08, $Q_{2,112[3]}$=5,58, $Q_{2,112[4]}$=3,08, $Q_{2,112[5]}$=0,58 dan $Q_{2,112[6]}$=-1,9.

Langkah 9 : $Q_{x,km[u]}>0$, Tidak. $N_{2,12[optimal]}$ =5 lanjut kelangkah 13.

Langkah 13: Set $o=\{m_1\}$ dengan s_1=45 dan t_1= 15, kembali kelangkah 7.

Langkah 7 : Set $N_{2,211}$=2.

Langkah 8 : Diperoleh ukuran *batch* $Q_{2,211[1]}$= 13 dan $Q_{2,211[2]}$=10.

Langkah 9 : $Q_{x,km[u]}>0$, ya.

Langkah 10 : Ya, berada pada operasi terakhir.

Langkah 12: Set $N_{2,211}$= 3, kembali ke langkah 8.

Langkah 8 : Diperoleh ukuran *batch* $Q_{2,211[1]}$= 10,67, $Q_{2,211[2]}$=7,67 dan $Q_{2,211[3]}$=4,66.

Langkah 9 : $Q_{x,km[u]}>0$, ya.

Langkah 10 : Ya, berada pada posisi terakhir

Langkah 12: Set $N_{2,211}$= 4, kembali ke langkah 8.

Langkah 8 : Diperoleh ukuran *batch* $Q_{2,211[1]}$= 10,25, $Q_{2,211[2]}$=7,25, $Q_{2,211[3]}$=4,25 dan $Q_{2,211[4]}$=1,25.

Langkah 9 : $Q_{x,km[u]}>0$, ya.

Langkah 10 : Ya, berada pada posisi terakhir.

Langkah 12: Set $N_{2,211}$= 5, kembali ke langkah 8.

Langkah 8 : Diperoleh ukuran *batch* $Q_{2,211[1]}$= 10,6, $Q_{2,211[2]}$=7,6, $Q_{2,211[3]}$=4,6, $Q_{2,211[4]}$=1,6 dan $Q_{2,211[5]}$=-1,4.

Langkah 9 : $Q_{x,km[u]}>0$, Tidak. $N_{2,21[optimal]}$ =4 lanjut kelangkah 13.

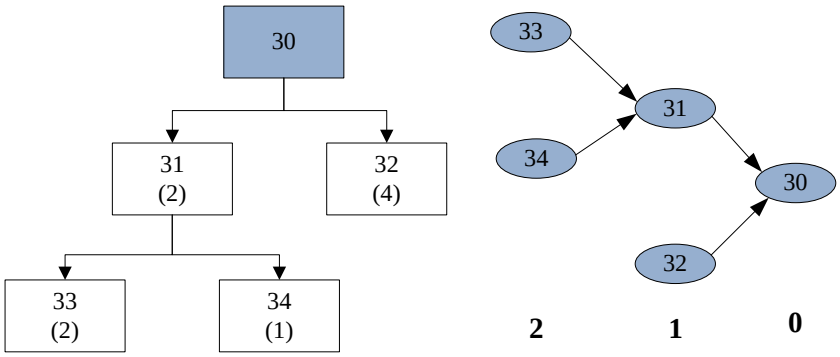
Langkah 13:Solusi optimal $N_{2,12}$= 5 ukuran *batch* $Q_{2,12[1]}$= 10,2, $Q_{2,12[2]}$=7,7, $Q_{2,12[3]}$=5,2, $Q_{2,12[4]}$=2,7 dan $Q_{2,12[5]}$=0,2. Serta $N_{2,21}$= 4 ukuran *batch* $Q_{2,21[1]}$= 10,25, $Q_{2,21[2]}$=7,25, $Q_{2,21[3]}$=4,25 dan $Q_{2,21[4]}$=1,25.

Langkah 14: Set $x=x+1=2+1=3$, $x>y$, Solusi optimal untuk setiap operasi komponen.

Komponen	Mesin	N	$Q_{ij[1]}$	$Q_{ij[2]}$	$Q_{ij[3]}$	$Q_{ij[4]}$	$Q_{ij[5]}$
P_{11}	4	3	22,67	12,67	2,66	-	-
	3	4	14,75	10,75	6,75	2,75	-
P_{12}	2	5	10,2	7,7	5,2	2,7	0,2
	1	4	10,25	7,25	4,25	1,25	-

Langkah 11 : Set α=2 untuk $P_{i0}=P_{30}$, kembali kelangkah 3.

Langkah 3 :



Langkah 4 : $P_{ij}=\{P_{31},P_{32},P_{33},P_{34}\}$, dengan $m = \{(m_3),(m_1),(m_3,m_4),(m_4,m_2)\}$

Langkah 5 :

P_{i0}	n_{i0}	$dr_{io}(\%)$	$n_{i0\ (diproduksi)}$	O_{i0km}	$s_{iokm\bullet}$	$t_{i0km\bullet}$	d_{i0}
P_{30}	20	8	22	O_{30102}	15	5	3300

Langkah 6 :

P_{ij}	$Z(P_{ij})$	H_{ij}
P_{31}	P_{30}	2
P_{32}	P_{30}	4
P_{33}	P_{31}	2
P_{34}	P_{31}	1

Langkah 7 :

Pij	n_{ij}	$dr_{ijkm\bullet}(\%)$	$n_{ij(diproduksi)}$	O_{ijklm}	$S_{ijkm\bullet}$	$t_{ijkm\bullet}$
P_{31}	44	8	48	O_{31113}	25	5
P_{32}	88	6	94	O_{32111}	30	10
P_{33}	106	5	112	O_{33123}	35	5
	96	10	106	O_{33224}	20	10
P_{34}	52	5	55	O_{34124}	45	20

	48	8	52	O_{34222}	60	15

Langkah 8 : Aturan *MiLF* dimulai dari level 0 (*l=0*).

Langkah 9 : Gunakan Sub Algoritma penentuan jumlah dan ukuran *batch*.

[Sub Algoritma Penentuan Jumlah dan Ukuran *Batch*]

Langkah 1 : $L(P_{ij})$ ={ P_{30}} dengan indeks x =1

Langkah 2 : $M(P_{30})$ = {m_2}

Langkah 3 : $N_{1,112}$=1 dengan $Q_{1,112[1]}$ = 22

Langkah 4: *Pilih* x=1

Langkah 5: Penjadwalan sesuai *routing* menghasilkan $F_{N=1}^{a} = 125$ Karena hanya diproses pada satu mesin,

Langkah 14: set x=x+1, maka didapat solusi optimal.

P_{i0}	N_1	$Q_{10102[N]}$
P_{30}	1	22

Langkah 10 : Set l=l+*1*=0+1=1, kembali ke langkah 9.

Langkah 9 : Gunakan Sub Algoritma penentuan jumlah dan ukuran *batch.*

[Sub Algoritma Penentuan Jumlah Batch]

Langkah 1 : $L(P_{ij})$ ={ P_{31}, P_{32}} dengan indeks x=1 dan x=2.

Langkah 2 : $M(P_{31})$ = { m_3} dan $M(P_{32})$ = { m_1}

Langkah 3 : $N_{1,113}$=1 dan $Q_{1,113}$ = 48

$N_{2,211}$=1 dan $Q_{2,211}$ = 94

Langkah 4 : Pilih komponen dengan indeks x =1.

Langkah 5 : Karena hanya diproses pada satu mesin maka didapatkan solusi optimal $N_{1,13}$=1 dan $Q_{1,13}$ = 48, dengan *flow time* $F_{N=1}^{a} = 265$

Langkah 13: Set x =2 dengan $N_{2,21}$=1 dan $Q_{2,21}$ = 94, kembali kelangkah 5.

Langkah 5 : Karena hanya diproses pada satu mesin maka didapatkan solusi optimal $N_{2,21}$=1 dan $Q_{2,21}$ = 94 dengan *flow time* $F_{N=1}^{a} = 970$

Langkah 14: Solusi optimal untuk langkah optimal untuk setiap komponen

Komponen	Mesin	N	$Q_{ij[1]}$
P_{31}	3	1	48
P_{32}	1	1	94

Langkah 10 : Set l= l+*1*=1+1=2, $l \leq L$, ya kembali kelangkah 9.

Langkah 9 : Gunakan Sub Algoritma penentuan jumlah dan ukuran *batch.*

[Sub Algoritma Penentuan Jumlah Batch]

Langkah 1 : $L(P_{ij})$ ={ P_{33}, P_{34}} dengan indeks x=1 dan x=2.

Langkah 2 : $M(P_{33})$ = {m_3, m_4} dan $M(P_{34})$ = {m_4, m_2}

Langkah 3 : $N_{1,123}$=1 dan $Q_{1,123}$=112

$N_{1,224}$=1 dan $Q_{1,224}$ = 106

$N_{2,124}$=1 dan $Q_{2,124}$= 55

$N_{2,222}$=1 dan $Q_{2,222}$= 52

Langkah 4 : Pilih komponen dengan indeks x=1

Langkah 5 : Penjadwalan sesuai *routing* menghasilkan

$F_{N=1}^{a} = 1655$

Langkah 6 : $M(P_{33})$={m_3} dengan s_{33123}=35 dan t_{33123}= 5

Langkah 7 : Set $N_{1,123}$=2

Langkah 8 : Diperoleh ukura *batch* $Q_{1,123[1]}$= 59,50 dan $Q_{1,123[2]}$=52.5.

Langkah 9 : $Q_{x,km[u]}>0$, ya.

Langkah 10 : Tidak, berada pada operasi pertama

Langkah 11 : Penjadwalan sesuai routing menghasilkan $F_{N=2}^{a} = 1392.5$ (1392.5≤1655)

Langkah 12 : Set $N_{1,123}$= 3, kembali ke langkah 8.

Langkah 8 : Diperoleh ukura *batch* $Q_{1,123[1]}$= 44.33, $Q_{1,123[2]}$=37.33 dan $Q_{1,123[3]}$=30.34

Langkah 9 : $Q_{x,km[u]}>0$, ya.

Langkah 10 : Tidak, berada pada operasi pertama

Langkah 11: Penjadwalan sesuai routing menghasilkan $F_{N=3}^{a} = 1316.65$ (1316.65≤1392.5)

Langkah 12: Set $N_{1,123}$= 4, kembali ke langkah 8.

Langkah 8 : Diperoleh ukura *batch* $Q_{1,123[1]}$= 38.5, $Q_{1,123[2]}$=31.5, $Q_{1,123[3]}$=24.5dan $Q_{1,123[4]}$=17.5.

Langkah 9 : $Q_{x,km[u]}>0$, ya.

Langkah 10 : Tidak, berada pada operasi pertama

Langkah 11 : Penjadwalan sesuai routing menghasilkan $F_{N=4}^{a} =$ 1287.5(1287.5≤1316.65)

Langkah 12: Set $N_{1,123}$= 5, kembali ke langkah 8.

Langkah 8 : Diperoleh ukura *batch* $Q_{1,123[1]}$= 36.4, $Q_{1,123[2]}$=29.40, $Q_{1,123[3]}$=22.40, $Q_{1,123[4]}$=15.40 dan $Q_{1,123[5]}$=8.40.

Langkah 9 : $Q_{x,km[u]}>0$, ya.

Langkah 10 : Tidak, berada pada operasi pertama

Langkah 11 : Penjadwalan sesuai routing menghasilkan $F^{a}_{N=5} = 1277$ (1277≤1287.5)

Langkah 12: Set $N_{1,123}$= 6, kembali ke langkah 8.

Langkah 8 : Diperoleh ukuran *batch* $Q_{1,123[1]}$= 36.17, $Q_{1,123[2]}$=29.17, $Q_{1,123[3]}$=22.17, $Q_{1,123[4]}$=15.17, $Q_{1,123[5]}$=8.17 dan $Q_{1,123[6]}$=1.16.

Langkah 9 : $Q_{x,km[u]}>0$, ya.

Langkah 10 : Tidak, berada pada operasi pertama

Langkah 11 : Penjadwalan sesuai routing menghasilkan $F^{a}_{N=5} =$ 1275.8(1275.8≤1277)

Langkah 12: Set $N_{1,123}$= 7, kembali ke langkah 8.

Langkah 8 : Diperoleh ukuran *batch* $Q_{1,123[1]}$= 37, $Q_{1,123[2]}$=30, $Q_{1,123[3]}$=23, $Q_{1,123[4]}$=16, $Q_{1,123[5]}$=9, $Q_{1,123[6]}$=2. dan $Q_{1,123[7]}$=-5.

Langkah 9 : $Q_{x,km[u]}>0$, Tidak. $N_{1,123[optimal]}$ = 5 lanjut ke langkah 13.

Langkah 13: Set $o=\{m_4\}$ dengan s_{33224}=20 dan t_{33224}= 10, kembali ke langkah 7.

Langkah 7 : Set $N_{1,224}$=2.

Langkah 8 : Diperoleh ukura *nbatch* $Q_{1,224[1]}$= 55.5 dan $Q_{1,224[2]}$=50.5.

Langkah 9 : $Q_{x,km[u]}>0$, ya.

Langkah 10 : Ya, berada pada operasi terakhir

Langkah 12: Set $N_{1,224}$= 3, kembali ke langkah 8.

Langkah 8 : Diperoleh ukuran *batch* $Q_{1,224[1]}$= 4033, $Q_{1,224[2]}$=35.33 dan $Q_{1,224[3]}$=30.33.

Langkah 9 : $Q_{x,km[u]}>0$, ya.

Langkah 10 : Ya, berada pada operasi terakhir

Langkah 12: Set $N_{1,224}$= 4, kembali ke langkah 8.

Langkah 8 : Diperoleh ukuran *batch* $Q_{1,224[1]}$= 34, $Q_{1,224[2]}$=29, $Q_{1,224[3]}$=24 dan $Q_{1,224[4]}$=19.

Langkah 9 : $Q_{x,km[u]}>0$, ya.

Langkah 10 : Ya, berada pada operasi terakhir

Langkah 12: Set $N_{1,224}$= 5, kembali ke langkah 8.

Langkah 8 : Diperoleh ukura *batch* $Q_{1,224[1]}$= 31.2, $Q_{1,224[2]}$=26.2, $Q_{1,224[3]}$=21.2, $Q_{1,224[4]}$=16.20, dan $Q_{1,224[5]}$=11.20

Langkah 9 : $Q_{x,km[u]}>0$, ya.

Langkah 10 : Ya, berada pada operasi terakhir

Langkah 12: Set $N_{1,224}$= 6, kembali ke langkah 8.

Langkah 8 : Diperoleh ukura *batch* $Q_{1,224[1]}$= 30.17, $Q_{1,224[2]}$=25.17, $Q_{1,224[3]}$=20.17, $Q_{1,224[4]}$=16.17, $Q_{1,224[5]}$=10.17 dan $Q_{1,224[6]}$=5.1.

Langkah 9 : $Q_{x,km[u]}>0$, ya.

Langkah 10 : Ya, berada pada operasi terakhir

Langkah 12: Set $N_{1,224}$= 7, kembali ke langkah 8.

Langkah 8 : Diperoleh ukura *batch* $Q_{1,224[1]}$= 30.14, $Q_{1,224[2]}$=25.14, $Q_{1,224[3]}$=20.14, $Q_{1,224[4]}$=15.14, $Q_{1,224[5]}$=10.14, $Q_{1,224[6]}$=5.1dan $Q_{1,224[7]}$=0.13.

Langkah 9 : $Q_{x,km[u]}>0$, ya.

Langkah 10 : Ya, berada pada operasi terakhir

Langkah 12: Set $N_{1,224}$= 8, kembali ke langkah 8.

Langkah 8 : Diperoleh ukura *batch* $Q_{1,224[1]}$= 30.75, $Q_{1,224[2]}$=25.75, $Q_{1,224[3]}$=20.75, $Q_{1,224[4]}$=15.75, $Q_{1,224[5]}$=10.75, $Q_{1,224[6]}$=5.75, $Q_{1,224[7]}$=0.0.75 dan $Q_{1,224[8]}$= -4.25.

Langkah 9 : $Q_{x,km[u]}>0$, tidak. $N_{1,224[optimal]}$ =6 lanjut kelangkah 13.

Langkah 13: Solusi optimal, $N_{1,123}$ =5 berukuran *batch* $Q_{1,123[1]}$= 34.40, $Q_{1,123[2]}$=27.40, $Q_{1,123[3]}$=20.40, $Q_{1,123[4]}$=13.40 dan $Q_{1,123[5]}$=6.40. Serta $N_{1,224}$= 7 ukuran *batch* $Q_{1,224[1]}$= 30.14, $Q_{1,224[2]}$=25.14, $Q_{1,224[3]}$=20.14, $Q_{1,224[4]}$=15.14, $Q_{1,224[5]}$=10.14, $Q_{1,224[6]}$=5.1dan $Q_{1,224[7]}$=0.13

Langkah 14: Set x = 2, $N_{2,124}$=1 dan $Q_{2,124}$=55, $N_{2,124}$=1 dan $Q_{2,24}$=52

Langkah 5 : Penjadwalan sesuai routing menghasilkan

$$F_{N=1}^{a} = 1100$$

Langkah 6 : $M(P_{34})$={m_4} dengan s_{34124}=45 dan t_{34124}= 15

Langkah 7 : Set $N_{2,11}$=2

Langkah 8 : Diperoleh ukura *batch* $Q_{1,124[1]}$= 32 dan $Q_{1,124[2]}$=23.

Langkah 9 : $Q_{x,km[u]}>0$, ya.

Langkah 10 : Tidak, berada pada operasi pertama

Langkah 11 : Penjadwalan sesuai routing menghasilkan $F^a_{N=2}$ = 985(985≤1100)

Langkah 12 : Set $N_{2,124}$= 3, kembali ke langkah 8.

Langkah 8 : Diperoleh ukura *batch* $Q_{2,124[1]}$= 27.33, $Q_{2,124[2]}$=18.33 dan $Q_{2,124[3]}$=9.34

Langkah 9 : $Q_{x,km[u]}>0$, ya.

Langkah 10 : Tidak, berada pada operasi pertama

Langkah 11 : Penjadwalan sesuai routing menghasilkan $F^a_{N=3}$ = 961.65(961.5≤985)

Langkah 12 : Set $N_{2,124}$= 4, kembali ke langkah 8.

Langkah 8 : Diperoleh ukuran *batch* $Q_{2,124[1]}$= 27.3, $Q_{2,124[2]}$=18.3, $Q_{2,124[3]}$=9.25 dan $Q_{2,124[4]}$=0.25

Langkah 9 : $Q_{x,km[u]}>0$, ya.

Langkah 10 : Tidak, berada pada operasi pertama

Langkah 11 : Penjadwalan sesuai routing menghasilkan $F^a_{N=3}$ = 961.65(961.5≤985)

Langkah 12 : Set $N_{2,124}$= 5, kembali ke langkah 8.

Langkah 8 : Diperoleh ukuran *batch* $Q_{2,124[1]}$= 29, $Q_{2,124[2]}$=20, $Q_{2,124[3]}$=11, $Q_{2,124[4]}$=2 dan $Q_{2,124[5]}$=-7

Langkah 9 : $Q_{x,km[u]}>0$, tidak terpenuhi, $N_{1,124[optimal]}$ =4 lanjut kelangkah 13.

Langkah 13: Set $o=\{m_2\}$ dengan s_{34222}=60 dan t_{34222}= 15, kembali ke langkah 7.

Langkah 7 : Set $N_{1,222}$=2.

Langkah 8 : Diperoleh ukuran *batch* $Q_{1,222[1]}$=28 dan $Q_{1,222[2]}$=24.

Langkah 9 : $Q_{x,km[u]}>0$, ya.

Langkah 10 : Ya, berada pada operasi terakhir

Langkah 12: Set $N_{1,222}$= 3, kembali ke langkah 8.

Langkah 8 : Diperoleh ukuran *batch* $Q_{1,222[1]}=21.3$, $Q_{1,222[2]}=17.3$ dan $Q_{1,222[3]}=13.3$.

Langkah 9 : $Q_{x,km[u]}>0$, ya.

Langkah 10 : Ya, berada pada operasi terakhir

Langkah 12: Set $N_{1,222}$= 4, kembali ke langkah 8.

Langkah 8 : Diperoleh ukuran *batch* $Q_{1,222[1]}=19$, $Q_{1,222[2]}=15$, $Q_{1,222[3]}=11$ dan $Q_{1,222[4]}=7$.

Langkah 9 : $Q_{x,km[u]}>0$, ya.

Langkah 10 : Ya, berada pada operasi terakhir

Langkah 12: Set $N_{1,222}$= 5, kembali ke langkah 8.

Langkah 8 : Diperoleh ukuran *batch* $Q_{1,222[1]}=18.4$, $Q_{1,222[2]}=14.4$, $Q_{1,222[3]}=10.4$, $Q_{1,222[4]}=6.4$ dan $Q_{1,222[5]}=2.4$.

Langkah 9 : $Q_{x,km[u]}>0$, ya.

Langkah 10 : Ya, berada pada operasi terakhir

Langkah 12: Set $N_{1,222}$= 6, kembali ke langkah 8.

Langkah 8 : Diperoleh ukuran *batch* $Q_{1,222[1]}=18.7$, $Q_{1,222[2]}=14.7$, $Q_{1,222[3]}=10.7$, $Q_{1,222[4]}=6.67$, $Q_{1,222[5]}=2.67$ dan $Q_{1,222[6]}=-1.3$.

Langkah 9 : $Q_{x,km[u]}>0$, tidak terpenuhi. $N_{1,222[optimal]}$ =5 lanjut kelangkah 13.

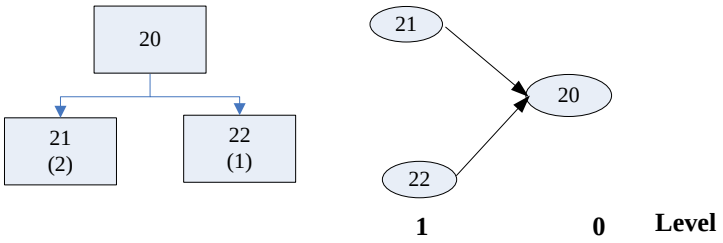
Langkah 13: Solusi optimal, $N_{1,124}$ =4 berukuran *batch* $Q_{2,124[1]}$= 27.3, $Q_{2,124[2]}$=18.3, $Q_{2,124[3]}$=9.25 dan $Q_{2,124[4]}$=0.2. Serta $N_{1,222}$= 5 ukuran *batch* $Q_{1,222[1]}=18.4$, $Q_{1,222[2]}=14.4$, $Q_{1,222[3]}=10.4$, $Q_{1,222[4]}=6.4$ dan $Q_{1,222[5]}=2.4$

Langkah 14: Set x = 2+1=3, maka

Komponen	Mesin	N	$Q_{ij[1]}$	$Q_{ij[2]}$	$Q_{ij[3]}$	$Q_{ij[4]}$	$Q_{ij[5]}$	$Q_{ij[6]}$	$Q_{ij[7]}$
P_{33}	3	6	36.17	29.17	22.17	15.17	8.17	1.17	-
	4	7	30.14	25.14	20.14	15.14	10.14	5.14	0.14
P_{34}	4	3	27.3	18.3	9.25	0.25	-	-	-
	2	5	18.4	14.4	10.4	6.4	2.4	-	-

Langkah 11 : Set α=3 untuk $P_{i0}=P_{20}$, kembali kelangkah 3.

Langkah 3 :



Langkah 4 : P_{ij}={P_{21},P_{22}}, dengan m = {(m_3,m_2),(m_1,m_4)}

Langkah 5 :

P_{i0}	n_{i0}	$dr_{io}(\%)$	$n_{i0\ (diproduksi)}$	O_{i0km}	S_{iokm}	t_{i0km}	d_{i0}
P_{20}	15	5	16	O_{20101}	40	15	3250

Langkah 6 :

P_{ij}	$Z(P_{ij})$	H_{ij}
P_{21}	P_{20}	2
P_{22}	P_{20}	1

Langkah 7 :

P_{ij}	n_{ij}	$dr_{ijkm}(\%)$	$n_{ij(diproduksi)}$	O_{ijklm}	$s_{ijkm\bullet}$	$t_{ijkm\bullet}$
P_{21}	34	8	37	O_{21113}	15	5
	32	6	34	O_{21212}	45	15
P_{22}	18	5	19	O_{22111}	100	25
	16	7	18	O_{22214}	50	15

Langkah 8 : *MiLF* level 0 (l=0).

Langkah 9 : Gunakan Sub Algoritma Penentuan jumlah dan ukuran batch.

[Sub Algoritma Penentuan Jumlah *Batch*]

Langkah 1: $L(P_{ij})$ ={ P_{20}} dengan indeks x=1

Langkah 2: $M(P_{20})$ = {m_1}

Langkah 3: $N_{1,101}$=1 dengan $Q_{1,101[1]}$ = 16

Langkah 4: *Pilih* x=1

Langkah 5: Penjadwalan sesuai *routing* menghasilkan $F^a_{N=1} = 280$ Karena hanya diproses pada satu mesin,

Langkah 14: set x=x+1= 1+1=2, x>y, maka didapat solusi optimal.

P_{i0}	N_1	$Q_{20[N]}$
P_{20}	1	16

Langkah 10 : set l=l+1= 0+1=1, kembali kelangkah 9.

Langkah 9 : Gunakan Sub Algoritma ukuran batch dan jumlah batch.

Langkah 1 : $L(P_{ij}) = \{ P_{21}, P_{22}\}$ dengan indeks x=1 dan x=2.

Langkah 2 : $M(P_{21}) = \{m_3, m_2\}$ dan $M(P_{22}) = \{m_1, m_4\}$

Langkah 3 : $N_{1,113}$=1 dan $Q_{1,113}$=37

$N_{1,212}$=1 dan $Q_{1,212}$=34

$N_{2,111}$=1 dan $Q_{2,111}$=19

$N_{2,214}$=1 dan $Q_{2,214}$=18

Langkah 4 : Pilih komponen dengan indeks x=1

Langkah 5 : Penjadwalan sesuai *routing* menghasilkan

$F^a_{N=1} = 710$

Langkah 6 : $M(P_{21})=\{m_3\}$ dengan s_{2113}=15 dan t_{2113}= 5

Langkah 7 : Set $N_{1,13}$=2

Langkah 8 : Diperoleh ukura *batch* $Q_{1,113[1]}$= 20 dan $Q_{1,113[2]}$=17.

Langkah 9 : $Q_{x,klm[u]}>0$, ya.

Langkah 10 : Tidak, berada pada operasi pertama

Langkah 11 : Penjadwalan sesuai routing menghasilkan $F^a_{N=2} =$ 625(625≤710)

Langkah 12: Set $N_{1,13}$= 3.

Langkah 8 : Diperoleh ukura *batch* $Q_{1,113[1]}$= 15,33, $Q_{1,113[2]}$=12,33 dan $Q_{1,113[3]}$=9,34.

Langkah 9 : $Q_{x,klm[u]}>0$, ya.

Langkah 10 : Tidak, berada pada operasi pertama

Langkah 11 : Penjadwalan sesuai routing menghasilkan $F^a_{N=3} =$ 601.65(601.65≤625)

Langkah 12: Set $N_{1,113}$= 4.

Langkah 8 : Diperoleh ukuran *batch* $Q_{1,113[1]}$= 13,75, $Q_{1,113[2]}$=10,75, $Q_{1,113[3]}$=7,75 dan $Q_{1,113[4]}$=4,75.

Langkah 9 : $Q_{x,klm[u]}>0$, ya.

Langkah 10 : Tidak, berada pada operasi pertama

Langkah 11 : Penjadwalan sesuai routing menghasilkan $F^a_{N=4} = 593.75$ (593.75≤601.65)

Langkah 12: Set $N_{1,13}$= 5.

Langkah 8 : Diperoleh ukuran *batch* $Q_{1,113[1]}$= 13,4, $Q_{1,113[2]}$=10,4, $Q_{1,113[3]}$=7,4, $Q_{1,113[4]}$=4,4 dan $Q_{1,113[5]}$=1,4.

Langkah 9 : $Q_{x,klm[u]}>0$, ya.

Langkah 10 : Tidak, berada pada operasi pertama

Langkah 11 : Penjadwalan sesuai routing menghasilkan $F_{N=4}^{a} = 592$ (592≤593.75)

Langkah 10: Set $N_{1,113}$= 6.

Langkah 8 : Diperoleh ukura *batch* $Q_{1,113[1]}$= 13,66, $Q_{1,113[2]}$=10,66, $Q_{1,113[3]}$=7,66, $Q_{1,113[4]}$=4,66, $Q_{1,113[5]}$=1,66 dan $Q_{1,113[6]}$=-1,33.

Langkah 9 : $Q_{x,klm[u]}>0$, Tidak terpenuhi. $N_{1,113[optimal]}$ =5 lanjut ke langkah 13.

Langkah 13: Set $o=\{m_2\}$ dengan s_2=45 dan t_2= 15, kembali ke langkah 7.

Langkah 7 : Set $N_{1,212}$=2.

Langkah 8 : Diperoleh ukuran *batch* $Q_{1,212[1]}$= 18,5 dan $Q_{1,212[2]}$=15,5.

Langkah 9 : $Q_{x,klm[u]}>0$, ya.

Langkah 10 : Ya, berada pada operasi terakhir

Langkah 12: Set $N_{1,212}$= 3.

Langkah 8 : Diperoleh ukuran *batch* $Q_{1,212[1]}$= 14,33, $Q_{1,212[2]}$=11,33 dan $Q_{1,212[3]}$= 8,34.

Langkah 9 : $Q_{x,klm[u]}>0$, ya.

Langkah 10 : Ya, berada pada operasi terakhir

Langkah 12: Set $N_{1,22}$= 4.

Langkah 8: Diperoleh ukuran *batch* $Q_{1,22[1]}$=13,$Q_{1,222[2]}$=10, $Q_{1,222[3]}$=7dan $Q_{1,222[4]}$=4.

Langkah 9 : $Q_{x,klm[u]}>0$, ya.

Langkah 10 : Ya, berada pada operasi terakhir

Langkah 12: Set $N_{1,22}$= 5.

Langkah 8 : Diperoleh ukuran *batch* $Q_{1,212[1]}$= 12,8, $Q_{1,212[2]}$=9,8, $Q_{1,212[3]}$=6,8 ,$Q_{1,212[4]}$=3,8 dan $Q_{1,212[5]}$=0,8

Langkah 9 : $Q_{x,klm[u]}>0$, ya.

Langkah 10 : Ya, berada pada operasi terakhir

Langkah 12: Set $N_{1,212}$= 6.

Langkah 8 : Diperoleh ukuran *batch* $Q_{1,212[1]}$= 13,17, $Q_{1,212[2]}$=10,17, $Q_{1,212[3]}$=7,17 ,$Q_{1,212[4]}$=4,17, $Q_{1,212[5]}$=1,17 dan $Q_{1,212[6]}$=-1,83

Langkah 9 : $Q_{x,klm[u]}>0$, Tidak Terpenuhi. $N_{1,212[optimal]}$ =5 lanjut kelangkah 13.

Langkah 13: Solusi optimal, $N_{1,13}$ =5 berukuran *batch* $Q_{1,13[1]}$= 13,4, $Q_{1,13[2]}$=10,4, $Q_{1,113[3]}$=7,4, $Q_{1,113[4]}$=4,4 dan $Q_{1,113[5]}$=1,4 . Serta $N_{1,212}$= 5 ukuran *batch* $Q_{1,212[1]}$= 12,8, $Q_{1,212[2]}$=9,8, $Q_{1,212[3]}$=6,8 ,$Q_{1,212[4]}$=3,8 dan $Q_{1,212[5]}$=0,8

Langkah 14: Set x = 2, $N_{2,111}$=1 dan $Q_{2,111}$=19, $N_{2,214}$=1 dan $Q_{2,214}$=18

Langkah 5 : Penjadwalan sesuai routing menghasilkan $F^a_{N=1} = 845$

Langkah 6 : $M(P_{22})=\{m_1\}$ dengan s_1=100 dan t_1= 25

Langkah 7 : Set $N_{2,111}$=2

Langkah 8 : Diperoleh ukuran *batch* $Q_{2,111[1]}$= 11,5 dan $Q_{2,111[2]}$=7,5.

Langkah 9 : $Q_{x,klm[u]}>0$, ya.

Langkah 10 : Tidak, berada pada operasi pertama

Langkah 11 : Penjadwalan sesuai routing menghasilkan $F^a_{N=2} = 657.5$ (657.5≤845)

Langkah 12: Set $N_{2,111}$= 3.

Langkah 8 : Diperoleh ukuran *batch* $Q_{2,111[1]}$= 10,33, $Q_{2,111[2]}$=6,33dan $Q_{2,111[3]}$=2,33.

Langkah 9 : $Q_{x,klm[u]}>0$, ya.

Langkah 10 : Tidak, berada pada operasi pertama

Langkah 11 : Penjadwalan sesuai routing menghasilkan $F^a_{N=2} = 628.5$ (628.5≤657.5)

Langkah 12: Set $N_{2,11}$= 4.

Langkah 8 : Diperoleh ukuran *batch* $Q_{2,111[1]}$= 10,75, $Q_{2,111[2]}$=6,75, $Q_{2,111[3]}$=2,75 dan $Q_{2,111[4]}$=-1,25.

Langkah 9 : $Q_{x,klm[u]}>0$, Tidak terpenuhi. $N_{2,111[optimal]}$ =3 lanjut kelangkah 13.

Langkah 13: Set $o=\{m_4\}$ dengan s_2=50 dan t_2= 15, kembali kelangkah 7.

Langkah 7 : Set $N_{2,214}$=2.

Langkah 8 : Diperoleh ukuran *batch* $Q_{2,214[1]}$= 10,67 dan $Q_{2,214[2]}$=7,33.

Langkah 9 : $Q_{x,klm[u]}>0$, ya.

Langkah 10 : Ya, berada pada operasi terakhir

Langkah 12: Set $N_{2,214}$= 3.

Langkah 8 : Diperoleh ukuran *batch* $Q_{2,214[1]}$= 9,33, $Q_{2,214[2]}$=6dan $Q_{2,214[3]}$=2,67.

Langkah 9 : $Q_{x,km[u]}>0$, ya.

Langkah 10 : Ya, berada pada operasi terakhir

Langkah 12: Set $N_{2,214}$= 4.

Langkah 8 : Diperoleh ukuran *batch* $Q_{2,24[1]}$= 9,5, $Q_{2,24[2]}$=6, 17, $Q_{2,24[3]}$=2,83 dan $Q_{2,24[4]}$=-0,5.

Langkah 9 : $Q_{x,km[u]}>0$, Tidak. $N_{2,24[optimal]}$ =3 lanjut kelangkah 13.

Langkah 13: Solusi optimal $N_{2,11}$= 3 ukuran *batch* $Q_{2,11[1]}$= 10,33, $Q_{2,11[2]}$=6,33dan $Q_{2,11[3]}$=2,33. Serta $N_{2,24}$= 3 Ukuran *batch* $Q_{2,24[1]}$= 9,33, $Q_{2,24[2]}$=6dan $Q_{2,24[3]}$=2,67.

Langkah 14: Set $x=x+1=2+1=3$, $x>y$. Solusi optimal untuk setiap operasi komponen.

Komponen	Mesin	*N*	$Q_{ij[1]}$	$Q_{ij[2]}$	$Q_{ij[3]}$	$Q_{ij[4]}$	$Q_{ij[5]}$
P_{21}	3	5	13,4	10,4	7,4	4,4	1,4
	2	5	12,8	9,8	6,8	3,8	0,8
P_{22}	1	3	10,33	6,33	2,33	-	-
	4	3	9.33	6	2,67	-	-

Langkah 11: Set α=4 >3, maka lanjut ke langkah 12.

Langkah 12: Penjadwalan secara *backward* dimulai dari *due date* terbesar.

Langkah 13 : $m_1=\{o_{20101},o_{12211},o_{22111},o_{32111}\}$; $m_2=\{o_{30102},o_{12112},o_{21212},o_{34222}\}$; $m_3=\{o_{10113},o_{11213},o_{21113},o_{31113,}\ o_{33123}\}$ dan $m_4=\{o_{11114},o_{22214},\ o_{33224}\ ,o_{34124}\}$

Langkah 14: Untuk setiap mesin lakukan langkah 15-28, dipilih m_1.

Langkah 15:$N_{ij,klm}$=1 dengan $Q_{20,101}$=15, $Q_{12,211}$=23, $Q_{22,111}$=19 dan $Q_{32,111}$=94.

Langkah 16: Didapat { $o_{20101},o_{12211},\ o_{32111}$}

Langkah 17: terdapat lebih dari satu operasi

Langkah 18: Indeks produk akhir berbeda.

Langkah 19: Didapat urutan { $o_{12211},\ o_{32111},o_{20101},$ } dengan α=1, α=2 dan α=3.

Langkah 20 : Mulai dari α=1yaitu{ o_{12211}}

Langkah 21 : Hanya terdapat satu operasi, maka dilakukan pemecahan *batch* O_{12211} menjadi 4 buah *batch* berukuran $Q_{12211[1]}$= 10,25, $Q_{12211[2]}$=7,25, $Q_{12211[3]}$=4,25 dan $Q_{12211[4]}$=1,25. Dengan menggunakan Teorema 2.1 diperoleh urutan pemrosesan batch secara *backward* pada m_1 sebagai berikut: $L_{1221[1]}$, $L_{1221[2]}$, $L_{1221[3]}$, dan $L_{12211[4]}$. Lanjut ke langkah 28.

Langkah 28: Ulangi langkah 16.

Langkah 16:Didapat{ o_{20101} , o_{32111}}.

Langkah 17: terdapat lebih dari satu operasi.

Langkah 18: Indeks produk akir berbeda

Langkah 19: Didapat urutan { o_{32111} ,o_{20101}, } dengan α=1dan α=2

Langkah 20: Mulai dari α=1yaitu{ o_{32111} }.

Langkah 21: Hanya terdapat satu operasi dan karena komponen P_{32} hanya diproses pada satu buah mesin maka operasi O_{32111} tetap dalam satu *batch* berukuran Q_{32111}=94, sehingga didapat urutan pemprosesan *batch* secara *backward* pada m_1 sebagai berikut : $L_{12211[1]}$, $L_{12211[2]}$, $L_{12211[3]}$, $L_{12211[4]}$ dan L_{32111}. Lanjut kelangkah 28.

Langkah 28: Ulangi langkah 16.

Langkah 16: Didapat { o_{20101}}

Langkah 17 : Hanya terdapat satu operasi dan karena komponen P_{20} hanya diproses pada satu mesin, maka operasi O_{2011} tetap dalam satu *batch* berukuran Q_{2011}=16. Sehingga didapatkan urutan pemrosesan *batch* secara *backward* pada $m_1$sebagai berikut $L_{12211[1]}$, $L_{12211[2]}$, $L_{12211[3]}$, $L_{12211[4]}$, L_{32111} dan L_{20101}. Lanjut kelangkah 28.

Langkah 16: Didapat { o_{22111} }

Langkah 17: Hanya terdapat satu operasi, maka dilakukan pemecahan *batch* O_{22111} kedalam 3 *batch* berukuran $Q_{22111[1]}$= 10,33, $Q_{22111[2]}$=6,33dan $Q_{22111[3]}$=2,33. Dengan Teorema 2.1 didapat urutan pemrosesan *batch* secara *backward* $L_{22111[1]}$, $L_{22111[2]}$, $L_{22111[3]}$. Sehingga urutan pemrosesan pada m_1 sebagai berikut $L_{12211[1]}$, $L_{12211[2]}$, $L_{12211[3]}$, $L_{12211[4]}$, L_{32111}, L_{20101}, $L_{22111[1]}$, $L_{22111[2]}$ dan $L_{22111[3]}$.

Langkah 14: Untuk tiap mesin lakukan langkah 15-28, pilih m_2.

Langkah 15: N_{ij}=1 dengan $Q_{30,102}$=22, $Q_{12,112}$=26, $Q_{21,212}$=34 dan $Q_{34,222}$= 52.

Langkah 16: Didapat {o_{30102},o_{12112},o_{21212}}

Langkah 17: Terdapat lebih dari operasi.

Langkah 18: Indeks produk berbeda.

Langkah 19: Didapat urutan {o_{12112}, o_{30102},o_{21212} } dengan α=1, α=2 dan α=3.

Langkah 20: Mulai dari α=1 yaitu{ o_{12112} }

Langkah21:Hanya terdapat satu operasi, maka dilakukan pemecahan *batch* operasi O_{1212} dipecah kedalam 5 buah *batch* berukuran *batch* $Q_{1212[1]}$= 10,2, $Q_{1212[2]}$=7,7, $Q_{1212[3]}$=5,2, $Q_{1212[4]}$=2,7 dan $Q_{1212[5]}$=0,2. Dengan menggunakan teorema 2.1 didapat urutan pemrosesan secara *backward* pada m_2 sebagai berikut $L_{1212\ [1]}$, $L_{1212\ [2]}$, $L_{1212\ [3]}$, $L_{1212\ [4]}$, dan $L_{1212[5]}$. Lanjut kelangkah 28.

Langkah 28: Ulangi langkah 16.

Langkah 16: Didapat {o_{30102} ,o_{21212} }

Langkah 17: Terdapat lebih dari satu operasi.

Langkah 18: Indeks produk berbeda.

Langkah 19: Didapat urutan {o_{30102} ,o_{21212} }dengan α=1 dan α=2.

Langkah 20: Mulai dari α=1 yaitu{o_{30102}}

Langkah 21: Hanya terdapat satu operasi, karena P_{30} hanya diproses pada satu mesin. Maka tetap dalam satu buah *batch* berukuran Q_{30102}=22. Sehingga didapat urutan pemroses pada m_2 sebagai berikut: $L_{12112\ [1]}$, $L_{12112\ [2]}$, $L_{12112\ [3]}$, $L_{12112\ [4]}$, $L_{12112[5]}$ dan L_{30102}. Lanjut ke langkah 28.

Langkah 28: Ulangi langkah 16.

Langkah 16: Didapat {$o_{21212,}$ o_{34222}}

Langkah 17: Terdapat lebih dari satu operasi.

Langkah 18: Indeks produk berbeda.

Langkah 19: Didapat urutan { $o_{21212,}$ o_{34222}}dengan α=1 dan α=2.

Langkah 20: Mulai dari α=1 yaitu{ $o_{21212,}$}

Langkah 21: Terdapat satu operasi, maka dilakukan pemecahan *batch* O_{21212} menjadi 5 buah *batch* berukuran $Q_{21212[1]}$= 12,8, $Q_{21212[2]}$=9,8, $Q_{21212[3]}$=6,8 ,$Q_{21212[4]}$=3,8 dan $Q_{1,212[5]}$=0,8 dengan menggunakan teorema 2.1 didapat urutan pemrosesan *batch* secara *backward* pada

berikut: $L_{34222\ [1]}$, $L_{34222\ [2]}$, $L_{34222\ [3]}$, $L_{34222\ [4]}$, $L_{34222\ [5]}$. Sehingga urutan pemrosesan pada m_2 sebagai berikut: $L_{12112\ [1]}$, $L_{12112\ [2]}$, $L_{12112\ [3]}$, $L_{12112\ [4]}$, $L_{12112[5]}$,L_{30102}, $L_{21212\ [1]}$, $L_{21212\ [2]}$, $L_{21212\ [3]}$, $L_{21212\ [4]}$, dan $L_{21212[5]}$

Langkah 16: didapat { o_{34222}}

Langkah17:Terdapat satu operasi, maka dilakukan pemecahan *batch* O_{34222} menjadi 5 buah *batch* berukuran $Q_{34222\ [1]}=18.4$, $Q_{34222\ [2]}=14.4$, $Q_{34222\ [3]}=10.4$, $Q_{34222\ [4]}=6.4$ dan $Q_{34222[5]}=2.4$dengan menggunakan teorema 2.1 didapat urutan pemrosesan *batch* secara *backward* pada berikut: $L_{34222\ [1]}$, $L_{34222\ [2]}$, $L_{34222\ [3]}$, $L_{34222\ [4]}$, $L_{34222\ [5]}$. Sehingga urutan pemrosesan pada m_2 sebagai berikut: $L_{12112\ [1]}$, $L_{12112\ [2]}$, $L_{12112\ [3]}$, $L_{12112\ [4]}$, $L_{12112[5]}$, L_{30102}, ,$L_{21212\ [1]}$, $L_{21212\ [2]}$, $L_{21212\ [3]}$, $L_{21212\ [4]}$, $L_{21212[5]}$,$L_{34222\ [1]}$, $L_{34222\ [2]}$, $L_{34222\ [3]}$, $L_{34222\ [4]}$, $L_{34222\ [5]}$. Lanjut kelangkah 28.

Langkah 14: Untuk tiap mesin lakukan langkah 15-28, pilih m_3.

Langkah 15: N_{ij}=1 dengan $Q_{10,103}$=11, $Q_{11,213}$=35, $Q_{21,113}$=37, $Q_{31,113}$=48 dan $Q_{32,123}$=106 .

Langkah 16: Didapat {o_{10103},o_{21113},o_{31113}}

Langkah 17: Terdapat lebih dari satu operasi.

Langkah 18: Indeks produk akhir berbeda.

Langkah 19: Didapat urutan {o_{10103}, o_{31113},o_{21113}}dengan α=1, α=2 dan α=3.

Langkah 20: Mulai dari α=1 yaitu{o_{10103} }

Langkah 21:Hanya terdapat satu karena P_{10} hanya diproses pada satu mesin, maka O_{10103} tetap dalam satu *batch* berukuran Q_{10103}=11. Sehingga urutan pemrosesan *batch* pada m_3 sebagai berikut L_{10103}. Lanjut kelangkah 28.

Langkah 28: Ulangi langkah 16.

Langkah 16: Didapat {o_{11213}, o_{31113}, o_{21113}}.

Langkah 17: Terdapat lebih dari satu operasi.

Langkah 18: Indeks produk berbeda.

Langkah 19: Didapat urutan {o_{11213}, o_{31113}, o_{21113}} dengan α=1, α=2 dan α=3.

Langkah 20: Mulai dari α=1 yaitu{o_{11213}}.

Langkah 21:Hanya terdapat satu operasi, maka dilakukan pemecahan *batch* operasi O_{11213} kedalam 4 buah *batch* berukuran $Q_{11213[1]}$= 14.75, $Q_{11213[2]}$=10,75, $Q_{11213[3]}$=6,75 dan $Q_{11213[4]}$=2,75, dengan

menggunakan teorema 1 didapatkan urutan pemrosesan *batch* secara *backward* : $L_{11213\ [1]}$, $L_{11213\ [2]}$, $L_{11213\ [3]}$, dan $L_{11213\ [4]}$. Sehingga urutan pemrosesan pada m_3 sebagai berikut L_{1013}, $L_{11213\ [1]}$, $L_{11213\ [2]}$, $L_{11213\ [3]}$, dan $L_{11213\ [4]}$ Lanjut kelangkah 28.

Langkah 28: Ulangi langkah 16.

Langkah 16: Didapat{o_{31113}, o_{21113}}.

Langkah 17: terdapat lebih dari satu operasi.

Langkah 18: Indeks produk akhir berbeda.

Langkah 19: Didapat urutan {o_{31113}, o_{21113}} dengan α=1 dan α=2.

Langkah 20: Mulai dari α=1, yaitu{o_{31113}}.

Langkah 21: Hanya terdapat satu operasi, karena P_{31} hanya diproses pada satu mesin, maka operasi O_{3113} tetap dalam satu *batch* berukuran Q_{3113}=48. Sehingga urutan pemrosesan pada m_3 sebagai berikut L_{10103}, $L_{11213\ [1]}$, $L_{11213\ [2]}$, $L_{11213\ [3]}$,$L_{11213\ [4]}$ dan L_{31113}. Lanjut kelangkah 28.

Langkah 28: ulangi langkah 16.

Langkah 16: Didapat { o_{21113}, o_{33123}}.

Langkah 17: terdapat lebih dari satu operasi.

Langkah 18: Indeks produk akhir berbeda.

Langkah 19: Didapat urutan { o_{21113}, o_{33123} } dengan α=1 dan α=2.

Langkah 20: Mulai dari α=1, yaitu{ o_{21113}}.

Langkah 21: Hanya terdapat satu operasi, maka dilakukan pemecahan *batch* operasi O_{34222}menjadi 5 buah *batch* berukuran $Q_{21113[1]}$*=13.4,* $Q_{21113\ [2]}$*=10.4,* $Q_{21113\ [3]}$ *=74,*dan $Q_{21113[4]}$*=4.4*. Sehingga urutan pemprosesan pada m_3 sebagai berikut L_{10113}, $L_{11213\ [1]}$, $L_{11213\ [2]}$, $L_{11213\ [3]}$,$L_{11213\ [4]}$, L_{31113}, $L_{\ 21113\ [1]}$, $L_{21113\ [2]}$, $L_{21113[3]}$, dan $L_{21113[4]}$ Lanjut kelangkah 28.

Langkah 28: ulangi langkah 16.

Langkah 16:Didapat {o_{33123}}.

Langkah 17: Hanya terdapat satu operasi, maka dilakukan pemecahan *batch* operasi O_{33123} menjadi 6 buah *batch* berukuran *batch* $Q_{33123[1]}$=26.57, $Q_{33123\ [2]}$=29.17, $Q_{33123\ [3]}$=22.17, $Q_{33123\ [4]}$=15.17, $Q_{33123\ [5]}$=8.17 dan $Q_{33123\ [6]}$=1,17, dengan menggunakan teorema 2.1 didapat urutan pemrosesan *batch* secara *backward* menjadi $L_{21113\ [1]}$, $L_{21113\ [2]}$,

$L_{21113[3]}$,$L_{21113\ [4]}$ dan $L_{2113[5]}$. Sehingga urutan pemrosesan pada m_3 sebagai berikut L_{10113}, $L_{11213\ [1]}$, $L_{11213\ [2]}$, $L_{11213\ [3]}$,$L_{11213\ [4]}$, L_{31113}, ,$L_{21113\ [4]}$ dan $L_{33123\ [1]}$, $L_{33123\ [2]}$, $L_{33123\ [3]}$,$L_{33123\ [4]}$, $L_{21113[5]}$, $L_{33123\ [5]}$,$L_{21113\ [1]}$, $L_{21113\ [2]}$, $L_{21113[3]}$

Langkah 14: Untuk setiap mesin lakukan langkah 15-28, pilih m_4.

Langkah 15: N_{ij}=1dengan $Q_{11,114}$=38, $Q_{22,214}$=18, $Q_{33,224}$=106 dan $Q_{34,124}$=55.

Langkah 16: Didapat {o_{11114},o_{22214},o_{33224}}.

Langkah 17:Terdapat lebih dari satu operasi.

Langkah 18: Indeks produk akhir berbeda.

Langkah 19: Didapat urutan {o_{11114},o_{22214},o_{33224}, }dengan α=1,α=2 dan α=3.

Langkah 20: Mulai dari α=1, yaitu{o_{11114}}.

Langkah21:Hanya terdapat satu operasi, maka dilakukan pemecahan *batch* O_{11114} menjadi 3 *batch* berukuran $Q_{11114[1]}$= 22,67, $Q_{11114[2]}$=12,67, dan $Q_{11114[3]}$= 2,66, dengan menggunakan teorema 2.1 didapat urutan pemrosesan secara *backward* menjadi $L_{11114\ [1]}$, $L_{11114\ [2]}$, $L_{11114\ [3]}$. Sehingga urutan pemrosesan pada mesin m_4 sebagai berikut $L_{11114\ [1]}$, $L_{11114\ [2]}$, dan $L_{11114\ [3]}$.

Langkah 28: Ulangi langkah 16.

Langkah 16: Didapat { o_{22214}, o_{33224}, o_{34124}, }.

Langkah 17: terdapat lebih dari satu operasi.

Langkah 18: Indeks produk akhir sama.

Langkah 19: Didapat urutan { o_{22214}, o_{33224}, o_{34124}, }dengan α=1,α=2 dan α=3.

Langkah 20: Mulai dari α=1, yaitu{ o_{22214}}.

Langkah 21: Hanya terdapat satu operasi. Maka dilakukan pemecahan operasi O_{22214} kedalam 3 buah *batch* berukuran $Q_{22214[1]}$= 9,33,$Q_{22214[2]}$=6 dan $Q_{22214[3]}$=2, menggunakan teorema 2.1 didapat urutan pemprosesan secara *backward* menjadi $L_{33224\ [1]}$, $L_{33224\ [2]}$, $L_{33224\ [3]}$, $L_{33224\ [4]}$, $L_{33224\ [5]}$, $L_{33224\ [6]}$ dan $L_{33224\ [7]}$. Sehingga urutan pemprosesan pada mesin m_4 sebagai berikut $L_{11114\ [1]}$, $L_{11114\ [2]}$, $L_{11114\ [3]}$, $L_{22214\ [1]}$, $L_{22214\ [2]}$, dan $L_{22214\ [3]}$,

Langkah 28: Ulangi langkah 16.

Langkah 16: Didapat {o_{33224}, o_{34124},}.

Langkah 17: terdapat lebih dari satu operasi.

Langkah 18: Indeks produk akhir sama.

Langkah 19: Didapat urutan { $o_{33224,}o_{34124}$ } dengan α=1 dan α=2.

Langkah 20: Mulai dari α=1, yaitu{ o_{33224}}.

Langkah 21: Hanya terdapat satu operasi. Maka dilakukan pemecahan operasi O_{33224} kedalam 7 *batch* dengan ukuran $Q_{33224[1]}$= 30.14, $Q_{33224\ [2]}$=25.14, $Q_{33224\ [3]}$=20.14, $Q_{33224\ [4]}$=15.14, $Q_{33224\ [5]}$=10.14, $Q_{33224[6]}$=5.1 dan $Q_{33224[7]}$=0.14.menggunakan teorema 2.1 didapat urutan pemprosesan secara *backward* menjadi $L_{33224\ [1],}$ $L_{33224\ [2],}$ $L_{33224\ [3]}$, $L_{33224\ [4],}$ $L_{33224\ [5],}$ $L_{33224\ [6]}$ dan $L_{33224\ [7]}$. Sehingga urutan pemprosesan pada mesin m_4 sebagai berikut $L_{11114\ [1],}$ $L_{11114\ [2],}$ $L_{11114\ [3]}$, $L_{22214\ [1],}$ $L_{22214\ [2],}$ $L_{22214\ [3]}$, $L_{33224\ [1],}$ $L_{33224\ [2],}$ $L_{33224\ [3]}$, *dan* $L_{33224\ [4],}$ Lanjut kelangkah 28

Langkah 28: Ulangi langkah 16.

Langkah 16: Didapat{ o_{34124}}.

Langkah17:Hanya terdapat satu operasi, maka dilakukan pemecahan operasi O_{34124} kedalam 4 *batch* dengan ukuran $Q_{34124[1]}$= 27.3, $Q_{34124\ [2]}$=18.3, $Q_{34124\ [3]}$=9.25 dan $Q_{34124[4]}$=0.2, dengan menggunakan teorema 2.1 didapat urutan pemrosesan secara *backward* menjadi $L_{22214\ [1],}$ $L_{22214\ [2],}$ $L_{22214\ [3]}$. Sehingga urutan pemrosesan pada mesin m_4 sebagai berikut $L_{11114\ [1],}$ $L_{11114\ [2],}$ $L_{11114\ [3]}$, $L_{22214\ [1],}$ $L_{22214\ [2]}$ $L_{22214\ [3]}L_{33224\ [1],}$ $L_{33224\ [2],}$ $L_{33224\ [3]}$, $L_{33224\ [4],}$ $L_{33224\ [5],}$ $L_{33224\ [6]}$, $L_{33224[7]}$, $L_{34124[1],}$ $L_{34124[2],}$ $L_{34124[3]}$, dan $L_{34124[4],}$. Lanjut ke *Langkah 29*.

Langkah 29:Pengabungan jadwal untuk setiap mesin untuk membentuk S_{akhir} dengan menggunakan teknik insersi.

Langkah 30: m_1 = {o_{20101}} selesai pada t =3250

m_2 = {o_{30102}} selesai pada t =3300

m_3 = {o_{10103}} selesai pada t =3400

Langkah 31: m_1 = {o_{12211}, o_{32111}, o_{22111}} dengan indeks x = 1,2,3.

m_2 = {o_{12112}, $o_{21212,}o_{34222}$, } dengan indeks x = 1,2,3.

m_3 = {o_{11213}, $o_{31113,}$ o_{21113}, o_{33123} } dengan indeks x = 1,2,3, 4

m_4 = {o_{11114}, $o_{22214,}$ o_{33224}, o_{34124} } dengan indeks x = 1,2,3,4.

Langkah 32 :Untuk x =1 didapat m_1 = {o_{12211}}, m_2 = {o_{12112}}, m_3 = {o_{11213}} dan m_4 = {o_{11114}}

Langkah 33: Untuk m_1 pada saat t = 3290 (o_{12211} harus diproses pada waktu t = *2945*-3290) ada operasi terjadwal {o_{20101}} (dari *t=3010* sampai *t=3250*), terdapat lebih dari satu mesin, maka o_{12211} pada m_{12} . Sehingga pada mesin m_{11} = {o_{20101}}, dan m_{12} = {o_{12211}}.

Untuk mesin m_2 pada saat t=3203 (o_{12112} harus diproses pada waktu t=2943 sampai t= 3203) terjadwal operasi {o_{30112}} (dari t=3190 sampat t=3300). Lanjut ke langkah 34.

Langkah 34 : Indeks produk akhir berbeda.

Langkah 37: Jika operasi {o_{12112}} ditempatkan didepan operasi {o_{30112}} akan menghasilkan waktu tinggal aktual sebesar 1190. Seandainya {o_{12112}}ditempatkan dibelakang {o_{30112}}, akan menghasilkan waktu tinggal aktual sebesar 835. Maka {o_{12112}} ditempatkang dibelakang{o_{30112}}.maka {o_{12112}} dijadwalkan selesai pada t=3175.

Untuk mesin m_3 pada t =3290 (o_{11213} harus diproses pada waktu t=3115 sampai t= 3290) terjadwal operasi {o_{10103}} (dari t=3290 sampat t=3400), terdapat lebih dari satu mesin, maka o_{11213} pada m_{32}. Sehingga pada mesin m_{31} = {o_{10103}}, dan m_{12} = {o_{11213}}.

Untuk mesin m_4 pada waktu t= 3150,34 (o_{11114} harus diproses dari t=3112,34 samapai t = 3150,34) tidak ada operasi terjadwal, maka {o_{11114}} dijadwalkan selesai pada t= 3150,34. Lanjut ke langkah 38

Langkah 38 :Tentukan x =2, didapat m_1 = {o_{32111}}, m_2 = {o_{21212}}, m_3 = {o_{31113}} dan m_4 = {o_{22214}}

Langkah 33 :Untuk m_1 pada saat t = 3190 (o_{32111} harus diproses pada waktu t=2250 sampai t= 3190) ada operasi terjadwal m_{11} = {o_{20101}}, dan m_{12} = {o_{12211}}, terdapat mesin m_{13} tersedia, maka operasi o_{32111} pada mesin m_{13}. Sehingga m_{11} = {o_{20101}}, m_{12} = {o_{12211}}dan m_{13} = {o_{3211}}.

Untuk m_2 pada saat t = 3010 (o_{21212} harus diproses pada waktu t=2840 sampai t= 3010) ada operasi terjadwal o_{12112} (dari t=2915 sampat t=3175).

Langkah 34 : Indeks produk akhir berbeda

Langkah 37: Jika operasi { o_{21212}} ditempatkan didepan operasi { o_{12112}} akan menghasilkan waktu tinggal aktual sebesar 2420. Seandainya { o_{21212}}ditempatkan dibelakang {o_{12112}}, akan menghasilkan waktu tinggal aktual sebesar 2405. Maka {o_{21212}} ditempatkang dibelakang{o_{12112}}.maka {o_{21212}} dijadwalkan selesai pada t=2890.

Untuk mesin m_3 pada saat t=3190 (o_{31113} harus diproses pada waktu t=2950 sampai t= 3190), tidak operasi terjadwal pada mesin m_{31}. Maka operasi { o_{31113}} dijadwalkan selesai pada t=3190.

Untuk mesin m_4 pada saat t=3010, tidak ada operasi terjadwal, maka operasi {o_{22214}} dijadwalkan selesai pada t=3010.

Langkah 38 :Tentukan x =3, didapat m_1 = {o_{22111}}, m_2 = {o_{34222}},
m_3 = {o_{31113}} dan m_4 = {o_{33224}}

Langkah 33 :

Untuk m_1 pada saat t=2855.05 (o_{22111} harus diproses pada waktu t=2380.05 sampai t= 2855.05) tidak ada operasi terjadwal pada mesin m_{11},m_{12},dan m_{13} maka{o_{22111}} ditempatkan pada m_{11}.

Untuk mesin m_2 pada saat t= 2950 (o_{34222} harus diproses pada waktu t=2170 sampai t= 2950) terdapat operasi terjadwal

Langkah 34: Indeks produk akhir berbeda berbeda

Langkah 37: Jika operasi { o_{34222}} ditempatkan didepan operasi { o_{21212}} akan menghasilkan waktu tinggal aktual sebesar 3265. Seandainya { o_{34222}} ditempatkan dibelakang operasi {o_{21212}} akan menghasilkan waktu tinggal aktual sebesar 3139.95. Sehingga operasi { o_{34222}} dijadwalkan mengikuti {o_{21212}, dan dijadwalkan selesai pada t=2335.

Untuk m_3 pada saat t= (o_{21113} harus diproses pada waktu t=2373 sampai t= 2558) tidak operasi terjadwal pada m_{31}, maka operasi {o_{21113}} dijadwalkan selesai pada t=2498.

Untuk mesin m_4 pada saat t=2950 (o_{33224} harus diproses pada waktu t=1890 sampai t= 2950) tidak operasi terjadwal pada m_{42}, maka operasi {o_{21113}} dijadwalkan m_{42} selesai pada t=2498.

Langkah 38 :Tentukan x =4, didapat m_3 = {o_{33123}},dan m_4 = {o_{34124}}

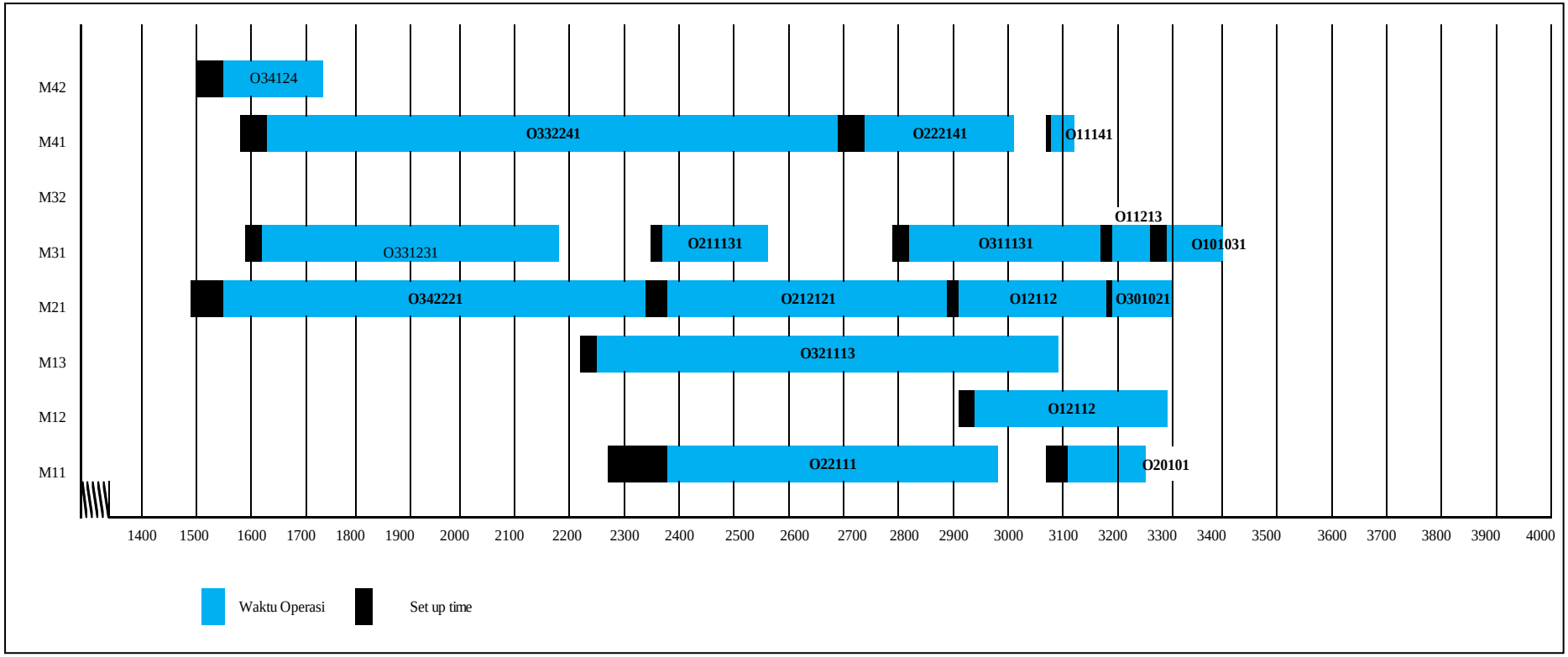
Langkah 33:

Untuk m_3 pada saat t=2444,15 (o_{33123} harus diproses pada waktu t=1884.5 sampai t= 2444.15) tidak ada operasi terjadwal pada m_{32}, maka operasi {o_{33123}} akan selesai pada t= 2444,15.

Untuk m_4 pada saat t=1553,75 (o_{34124} harus diproses pada waktu t=1278.75 sampai t= 1553,75) tidak ada operasi terjadwal pada m_{41}, maka operasi {o_{34124}} akan selesai pada t= 1553,75.

Langkah 38: *set* x =5, $x>y$ (5>4) untuk semua, lanjut ke langkah 39.

Langkah 39. Gambarkan *Gant Chart* dari S_{akhir} dengan memperhatikan ketersediaan mesin.



Gambar 1.2 Gant Chart Akhir Kondisi Statis

```
package formula;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;
import formula.model.Factory;
import formula.model.Formula;
import formula.model.Item;
import formula.model.Machine;
import formula.model.MachineOperation;
import formula.model.Operation;
import formula.model.Pair;
import formula.model.Target;

public class Pemenuhan {

	private HashMap<Integer, Machine> mapMachine = new
HashMap<Integer, Machine>();
	private HashMap<Integer, Item> mapItem = new HashMap<Integer,
Item>();

	public Item getItem(int id) {
		return mapItem.get(id);
	}

	public Machine getMachine(int id) {
		return mapMachine.get(id);
	}

	public Pemenuhan() {

	}

	private class OrderByDueDate implements Comparator<Target> {

		public static final int ASC = 1;
		public static final int DESC = 2;

		private int order = ASC;

		public OrderByDueDate(int order) {
			this.order = order;
		}

		@Override
		public int compare(Target t1, Target t2) {
			int v = t1.getDueDate() - t2.getDueDate();
			if(order==DESC) return -v;
			return v;
		}

	}
```

```
     public void case2() {
          init(new int[]{10,11,12,20,21,22,30,31,32,33,34},new
int[]{1,2,3,4},new int[]{3,1,2,2},true);
          Item item10 = getItem(10);
          Item item11 = getItem(11);
          Item item12 = getItem(12);
          Item item20 = getItem(20);
          Item item21 = getItem(21);
          Item item22 = getItem(22);
          Item item30 = getItem(30);
          Item item31 = getItem(31);
          Item item32 = getItem(32);
          Item item33 = getItem(33);
          Item item34 = getItem(34);

          item10.addChild(item11, 3);
          item10.addChild(item12, 2);

          item20.addChild(item21, 2);
          item20.addChild(item22, 1);

          item30.addChild(item31, 2);
          item30.addChild(item32, 4);

          item31.addChild(item33, 2);
          item31.addChild(item34, 1);

          item10.addFormula(getMachine(3),  30, 10,  4);
          item20.addFormula(getMachine(1),  40, 15,  5);
          item30.addFormula(getMachine(2),  15,  5,  8);
          item11.addFormula(getMachine(4),  10,  1,  6);
          item11.addFormula(getMachine(3),  20,  5,  5);
          item12.addFormula(getMachine(2),  25, 10, 10);
          item12.addFormula(getMachine(1),  45, 15,  4);
          item21.addFormula(getMachine(3),  15,  5,  8);
          item21.addFormula(getMachine(2),  45, 15,  6);
          item22.addFormula(getMachine(1), 100, 25,  5);
          item22.addFormula(getMachine(4),  50, 15,  7);
          item31.addFormula(getMachine(3),  25,  5,  8);
          item32.addFormula(getMachine(1),  30, 10,  6);
          item33.addFormula(getMachine(3),  35,  5,  5);
          item33.addFormula(getMachine(4),  50, 10, 10);
          item34.addFormula(getMachine(4),  45,  5,  5);
          item34.addFormula(getMachine(2),  60, 15,  8);


          List<Target> list = new ArrayList<Target>();
          list.add(new Target(item10, 10, 0,3400));
          list.add(new Target(item20, 15, 0,3250));
          list.add(new Target(item30, 20, 0,3300));

          Factory factory = new Factory();
          factory.setupMachine(mapMachine);
          factory = solve(0,factory,list);
          factory.debug();
```

```
          for(Operation o:factory.getListOrder())
               System.out.println(o);
          // System.out.println("WTA = " +
factory.calcWaktuTinggalAktual(list));
     }
private Factory solveBackward(Factory factory, List<Target>
listTarget) {


          // langkah 30 (menempatkan operasi terakhir pada produk
akhir)
          for(Target t:listTarget) {
               Item item = t.getItem();

               Operation oper = factory.findOperation(item,
item.getListFormula().size());
               if(oper.isVisited()) continue;
               oper.setTime(t.getDueDate()-oper.calcProcessTime(),
t.getDueDate());
               Machine m = oper.getFormula().getMachine();
               boolean result = factory.insertOperation(oper);
               factory.getListOrder().add(oper);
               oper.setVisited(true);
          }
          //System.out.println(mapTime);
          //System.out.println("WTA = " +
calcWaktuTinggalAktual(listTarget, mapTime));


          boolean running = true;
          long comb = 1;
          while(running) {
               running = false;
               for(Machine machine:factory.getListMachine()) {
                    for(Operation
operation:factory.getListOperation()) {

     if(operation.getFormula().getMachine()!=machine ||
operation.isVisited()) continue;
                         Operation parent = operation.getParent();
                         // System.out.println("OPER = " +
operation);
                         if(!parent.isVisited()) continue;

                         double dueDate = parent.getStartTime();
                         // cek batching
                         if(operation.getItem()==parent.getItem())
{
                              if(operation.getProcessTime() <=
parent.getProcessTime()) {
                                   double temp =
parent.getStartTime() - (operation.getLastBatch() *
operation.getProcessTime());
                                   dueDate = temp +
operation.calcProcessTime();
```

```
                                    }
                                    else {
                                          dueDate =
parent.getFinishTime() - (operation.getFirstBatch() *
parent.getProcessTime());
                                    }
                              }
                              operation.setTime(dueDate-
operation.calcProcessTime(),dueDate);

                              Factory bestFactory = null;


                              for(int
i=0;i<factory.getListMachineOperation().size();i++) {
                                    MachineOperation mo =
factory.getListMachineOperation().get(i);

     if(mo.getMachine()!=operation.getFormula().getMachine())
continue;
                                    for(int j=0;j<=mo.size();j++) {

                                          Factory tempFactory = new
Factory(factory);
                                          Operation tempOperation =
tempFactory.createOperation(operation);

     tempOperation.setVisited(true);

     tempFactory.getListMachineOperation().get(i).add(j,
tempOperation);

     tempFactory.getListOrder().add(tempOperation);
                                          //System.out.println("DEBUG =
" + tempOperation);

                                          //tempFactory.debug();
                                          boolean normal =
tempFactory.normalize(tempOperation);
                                          //System.out.println(normal +
" WTA  = " + tempFactory.calcWaktuTinggalAktual(listTarget));
                                          if(!normal) continue;
                                          else comb++;
                                          if(bestFactory == null ||
tempFactory.calcWaktuTinggalAktual(listTarget) <
bestFactory.calcWaktuTinggalAktual(listTarget))
                                                bestFactory =
tempFactory;
                                    }
                              }


                              if(bestFactory==null) return null;

                              factory = bestFactory;
```

```
     private Factory solveForward(Factory factory, List<Target>
listTarget) {

          OrderBySpecial orderBySpecial = new OrderBySpecial();
          List<Operation> temp2 = new
ArrayList<Operation>(factory.getListOperation());

          Collections.sort(factory.getListOperation(),
orderBySpecial);
          System.out.println(factory.getListOperation());

          for(int m=1;m<=mapMachine.size();m++) {
               System.out.println("m="+m);
               for(Operation o:factory.getListOperation()) {
                    if(o.getFormula().getMachine().getId()==m) {
                         System.out.println(o + " " +
factory.listChild(o));
                    }
               }
          }

          /*
          for(Target t:listTarget) {
               System.out.println(t.getItem());
               for(Operation o:factory.getListOperation()) {
                    if(o.getFinalItem()==t.getItem())
                         System.out.println("U " + o);
               }
          }
          */

          boolean running = true;
          long comb = 0;
          while(running) {
               running = false;
               for(Machine machine:factory.getListMachine()) {
                    for(Operation
operation:factory.getListOperation()) {

     if(operation.getFormula().getMachine()!=machine ||
operation.isVisited()) continue;
                         boolean ok =
factory.allChildVisited(operation);
                         if(!ok) continue;


     //System.out.println(operation.getItem().getLevel() + " OPER "
+ operation + " " + operation.length());
                         double start = 0;
                         for(Operation
child:factory.listChild(operation)) {

     if(child.getItem().getLevel()==operation.getItem().getLevel())
{
```

```
     if(child.getFormula().getProcessTime() <=
operation.getFormula().getProcessTime()) {
                                        double a =
operation.getFormula().getSetupTime();
                                        double b =
child.getFormula().getSetupTime() + child.getLastBatch() *
child.getFormula().getProcessTime();
                                        start =
(child.getStartTime() - child.getFormula().getSetupTime()) +
Math.max(a,b);
                                        // start -=
operation.getFormula().getSetupTime();
                                   }else {
                                        double endParent =
child.getFinishTime() + child.getFirstBatch() *
operation.getFormula().getProcessTime();
                                        start = endParent -
operation.calcProcessTime();
                                   }
                              }
                              else {
                                   start = Math.max(start,
child.getFinishTime());
                              }
                         }

                         double b =
factory.getArrivalTime(operation.getFinalItem(), listTarget);
                         if(b>0) {
                              start = Math.max(start,
b+operation.getFormula().getSetupTime());
                         }

                         // start +=
operation.getFormula().getSetupTime();
                         operation.setTime(start, start +
operation.calcProcessTime());
                         System.out.println("INSERT " +
operation);

                         operation.setVisited(true);

                         Factory bestFactory = null;


                         for(int
i=0;i<factory.getListMachineOperation().size();i++) {
                              MachineOperation mo =
factory.getListMachineOperation().get(i);

     if(mo.getMachine()!=operation.getFormula().getMachine())
continue;
                              for(int j=0;j<=mo.size();j++) {
```

```
     public Factory solve(double time,Factory factory,List<Target>
listTarget) {
          Collections.sort(listTarget,new
OrderByDueDate(OrderByDueDate.DESC));

          List<Item> q = new ArrayList<Item>();
          Map<Item, Integer> mapN = new LinkedHashMap<Item,
Integer>();

          for(Target t:listTarget) {
               if(t.getStartTime()==time && !t.isVisited()) {
                    t.setVisited(true);
                    q.add(t.getItem());
                    mapN.put(t.getItem(), t.getQuantity());
               }
          }
          while(!q.isEmpty()) {
               Item f = q.get(0); q.remove(0);

               List<Formula> listFormula = f.getListFormula();
               int n = mapN.get(f);
               for(int j=listFormula.size()-1;j>=0;j--) {
                    double e = listFormula.get(j).getError();
                    double p = 1.0 + (e/100.0);
                    n = (int)Math.ceil(n * p);

                    Operation operation = new Operation(f, j+1,
f.getListFormula().get(j),n);
                    Operation parent = null;
                    if(j==listFormula.size()-1) {
                         if(f.getParent()!=null) parent =
factory.findOperation(f.getParent(), 1);
                    }else {
                         parent = factory.findOperation(f, j+2);
                    }

                    operation.setParent(parent);
                    factory.getListOperation().add(operation);
               }
               mapN.put(f, n);

               for(Pair<Item, Integer> c:f.getListChild()) {
                    c.getFirst().setLevel(f.getLevel() + 1);
                    q.add(c.getFirst());
                    mapN.put(c.getFirst(), n * c.getSecond());
               }
          }
          factory.locking(time);
          for(Operation operation:factory.getListOperation()) {
               operation.solveBatch();
          }
          /*
          for(Operation operation:factory.getListOperation())
```

```
                System.out.println(operation.isLocked() + " "
+operation + " " + operation.toString2());
          */

          // return solveForward(factory, listTarget);
          return solveBackward(factory, listTarget);
     }

     public static void main(String []args) {
          long S = System.currentTimeMillis();
          Pemenuhan main = new Pemenuhan();
          main.case64();
          long F = System.currentTimeMillis();
          System.out.println("Computation Time = " + ((F-
S)/1000.0));
     }
}
```

n_{10}= 10; d_{10} =252000 (menit)

p_{ij}	Level	$Z(p_{ij})$	hij	Operation	Mesin	*Set-up time* (menit)	Waktu proses (menit)	Tingkat Cacat (%)
10	**0**		**1**	1	3	10	5	5
	1			1	2	15	1	10
11	**1**	**10**	**3**	2	1	20	10	7
	1			1	4	10	5	5
12	**1**	**10**	**1**	2	2	15	15	7
	2			1	4	25	10	4
13	**2**	**11**	**1**	2	3	35	15	8
	2			1	5	40	5	9
14	**2**	**11**	**3**	2	1	10	15	6
	2			1	3	20	10	4
15	**2**	**12**	**2**	2	1	25	1	5
	2			1	5	15	5	7
16	**2**	**12**	**3**	2	1	25	10	8
	3			1	3	40	5	6
17	**3**	**13**	**2**	2	4	30	15	9
	3			1	2	10	10	10
18	**3**	**13**	**1**	2	5	25	5	5
	3			1	2	30	1	5
19	**3**	**14**	**1**	2	4	15	10	4
	3			1	5	20	15	9
110	**3**	**14**	**3**	2	3	40	5	4
	3			1	2	60	15	6
111	**3**	**15**	**2**	2	3	50	10	7
	3			1	1	75	1	4
112	**3**	**15**	**3**	2	3	10	5	10
	3			1	2	15	10	7
113	**3**	**16**	**2**	2	4	20	5	6
	3			1	5	15	15	10
114	**3**	**16**	**1**	2	3	20	10	6
	4			1	2	25	5	9
115	**4**	**17**	**1**	2	4	30	10	6
	4			1	1	40	5	10
116	**4**	**17**	**3**	2	2	15	15	9
	4			1	5	10	10	6
117	**4**	**18**	**1**	2	1	15	15	5
118	**4**	**18**	**1**	1	1	20	5	8

p_{ij}	Level	$Z(p_{ij})$	hij	Operation	Mesin	*Set-up time* (menit)	Waktu proses (menit)	Tingkat Cacat (%)
	4			2	5	45	15	4
	4			1	2	30	10	4
119	**4**	**19**	**1**	2	5	15	1	8
	4			1	1	10	5	4
120	**4**	**19**	**3**	2	4	35	10	8
	4			1	3	20	5	6
121	**4**	**110**	**1**	2	1	35	15	5
	4			1	3	15	10	4
122	**4**	**110**	**1**	2	4	10	5	9
	4			1	4	50	1	7
123	**4**	**111**	**3**	2	5	45	10	8
	4			1	3	25	15	7
124	**4**	**111**	**2**	2	1	20	5	5
	4			1	2	15	15	4
125	**4**	**112**	**3**	2	4	10	10	7
	4			1	5	15	1	6
126	**4**	**112**	**1**	2	3	20	5	4
	4			1	5	15	10	7
127	**4**	**113**	**1**	2	4	20	15	6
	4			1	2	35	10	5
128	**4**	**113**	**2**	2	3	15	5	10
	4			1	1	10	1	5
129	**4**	**114**	**2**	2	3	20	10	9
	4			1	4	40	15	7
130	**4**	**114**	**3**	2	2	60	5	4

n_{20}= 20; d_{10} =250000 (menit)

p_{ij}	Level	$Z(p_{ij})$	hij	Operation	Mesin	*Set-up time* (menit)	Waktu proses (menit)	Tingkat Cacat (%)
20	**0**		**1**	1	2	15	15	8
	1			1	5	25	5	4
21	**1**	**20**	**1**	2	2	35	15	10
	1			1	3	40	10	7
22	**1**	**20**	**3**	2	2	10	5	5
23	**2**	**21**	**1**	1	2	25	10	8

(Lanjutan)

p_{ij}	Level	$Z(p_{ij})$	hij	Operation	Mesin	*Set-up time* (menit)	Waktu proses (menit)	Tingkat Cacat (%)
	2			2	4	30	15	5
	2			1	4	15	5	4
24	**2**	**21**	**2**	2	1	20	15	4
	2			1	2	40	10	5
25	**2**	**22**	**3**	2	4	60	20	8
	2			1	4	50	5	8
26	**2**	**22**	**3**	2	1	20	15	4
	3			1	2	25	10	10
27	**3**	**23**	**1**	2	1	30	1	9
	3			1	4	40	5	4
28	**3**	**23**	**3**	2	5	15	10	4
	3			1	4	10	15	6
29	**3**	**24**	**1**	2	2	15	25	7
	3			1	4	20	15	8
210	**3**	**24**	**3**	2	2	40	5	10
	3			1	5	60	10	9
211	**3**	**25**	**3**	2	2	50	15	10
	3			1	1	75	5	6
212	**3**	**25**	**3**	2	4	10	20	6
	3			1	4	15	10	4
213	**3**	**26**	**3**	2	1	20	1	6
	3			1	3	15	5	6
214	**3**	**26**	**3**	2	5	20	10	9
	4			1	3	25	5	4
215	**4**	**27**	**4**	2	2	30	10	8
	4			1	3	40	5	8
216	**4**	**27**	**1**	2	5	15	15	10
	4			1	1	10	10	10
217	**4**	**28**	**2**	2	2	15	15	6
	4			1	3	20	5	10
218	**4**	**28**	**2**	2	1	45	15	10
	4			1	2	30	10	9
219	**4**	**29**	**3**	2	5	35	1	6
	4			1	2	15	5	8
220	**4**	**29**	**4**	2	3	10	10	6
	4			1	5	20	5	7
221	**4**	**210**	**3**	2	4	35	15	6

p_{ij}	Level	$Z(p_{ij})$	hij	Operation	Mesin	*Set-up time* (menit)	Waktu proses (menit)	Tingkat Cacat (%)
	4			1	3	15	10	9
222	**4**	**210**	**4**	2	1	10	10	5
	4			1	2	50	5	6
223	**4**	**211**	**4**	2	5	20	15	9
	4			1	4	25	10	6
224	**4**	**211**	**2**	2	1	30	5	9
	4			1	5	40	10	9
225	**4**	**212**	**1**	2	1	15	5	8
	4			1	5	10	15	7
226	**4**	**212**	**3**	2	2	15	10	5
	4			1	5	20	15	6
227	**4**	**213**	**4**	2	4	20	5	10
	4			1	3	35	15	5
228	**4**	**213**	**1**	2	5	15	10	6
	4			1	2	10	1	9
229	**4**	**214**	**4**	2	1	20	5	4
	4			1	2	40	10	7
230	**4**	**214**	**1**	2	3	60	5	8

n_{30}= 15; d_{10} =251000 (menit)

p_{ij}	Level	$Z(p_{ij})$	hij	Operation	Mesin	*Set-up time* (menit)	Waktu proses (menit)	Tingkat Cacat (%)
30	**0**		**1**	1	4	30	5	6
	1			1	5	15	15	9
31	**1**	**30**	**3**	2	2	20	5	7
	1			1	4	40	15	6
32	**1**	**30**	**2**	2	3	60	10	10
	2			1	3	50	20	7
33	**2**	**31**	**3**	2	5	20	5	10
	2			1	3	25	15	4
34	**2**	**31**	**1**	2	5	30	10	8
	2			1	5	40	1	4
35	**2**	**32**	**2**	2	1	60	5	4
	2			1	4	50	10	6
36	**2**	**32**	**3**	2	2	20	15	5

(Lanjutan)

p_{ij}	Level	$Z(p_{ij})$	hij	Operation	Mesin	*Set-up time* (menit)	Waktu proses (menit)	Tingkat Cacat (%)
	3			1	3	25	25	7
37	**3**	**33**	**3**	2	2	15	15	9
	3			1	2	10	5	10
38	**3**	**33**	**2**	2	4	15	10	6
	3			1	5	20	15	9
39	**3**	**34**	**1**	2	3	45	5	10
	3			1	3	30	20	7
310	**3**	**34**	**3**	2	1	35	10	10
	3			1	4	15	1	7
311	**3**	**35**	**2**	2	1	10	5	6
	3			1	3	75	5	5
312	**3**	**35**	**3**	2	1	10	20	9
	3			1	4	15	10	9
313	**3**	**36**	**2**	2	1	20	1	10
	3			1	2	15	5	9
314	**3**	**36**	**1**	2	3	20	10	5
	4			1	3	25	5	9
315	**4**	**37**	**3**	2	1	30	10	8
	4			1	3	40	5	9
316	**4**	**37**	**2**	2	5	15	15	9
	4			1	1	10	10	6
317	**4**	**38**	**2**	2	2	15	15	4
	4			1	5	20	5	8
318	**4**	**38**	**2**	2	1	45	15	9
	4			1	1	30	10	10
319	**4**	**39**	**3**	2	2	35	1	4
	4			1	1	15	5	8
320	**4**	**39**	**2**	2	2	10	10	5
	4			1	1	20	5	10
321	**4**	**310**	**2**	2	3	35	15	10
	4			1	5	15	10	9
322	**4**	**310**	**2**	2	4	10	10	9
	4			1	4	50	5	7
323	**4**	**311**	**3**	2	3	20	15	4
	4			1	1	25	10	6
324	**4**	**311**	**3**	2	5	15	20	6
	4			1	5	10	10	8
325	**4**	**312**	**2**	2	1	15	1	7

(Lanjutan)

p_{ij}	Level	$Z(p_{ij})$	hij	Operation	Mesin	*Set-up time* (menit)	Waktu proses (menit)	Tingkat Cacat (%)
	4			1	3	20	5	7
326	**4**	**312**	**1**	2	2	45	5	10
	4			1	3	30	20	6
327	**4**	**313**	**2**	2	5	35	10	10
	4			1	3	15	1	10
328	**4**	**313**	**2**	2	4	35	5	5
	4			1	5	15	10	8
329	**4**	**314**	**1**	2	3	10	5	10
330	**4**	**314**	**2**	1	4	40	10	6