BAB 2
LANDASAN TEORI

2.1 Penjadwalan Tugas Misdinar

Berdasarkan KBBI, misdinar diartikan sebagai pemuda atau pemudi yang melayani pastor yang memimpin upacara Gereja Katolik seperti misa atau ibadat [7]. Pada kasus Gereja Santa Bernadet Paroki Ciledug, seorang pemuda atau pemudi yang beragama Kristen Katolik dapat menjadi seorang misdinar dengan memenuhi syarat-syarat Gereja tertentu dan mengikuti pelatihan yang diadakan oleh organisasi misdinar. Pelatihan tersebut bertujuan untuk mengasah kemampuan calon misdinar supaya dapat menjalankan tugas-tugas dalam melayani misa atau ibadat dengan baik. Setelah dipastikan memenuhi syarat dan lulus pelatihan, maka seorang pemuda atau pemudi Katolik tersebut dapat dilantik menjadi misdinar.

Misdinar yang sudah dilantik kemudian dapat ditugaskan untuk melayani misa atau ibadat yang sesungguhnya. Supaya semua anggota misdinar memperoleh kesempatan yang sama untuk bertugas, maka setiap anggota misdinar harus saling bergiliran dengan anggota misdinar lainnya. Untuk itu, dibuatlah jadwal tugas misdinar yang dapat menjamin pergiliran tugas tersebut.

Karena sampai saat ini gedung gereja resmi dari Gereja Santa Bernadet Paroki Ciledug masih dalam proses pembangunan, sementara ini misa atau ibadat diadakan pada 2 tempat yang berada di daerah yang berbeda: Pinang dan Metro Permata [11]. Terdapat kemungkinan terjadi perubahan terhadap pengadaan misa di kedua daerah tersebut jika nantinya Gereja Santa Bernadet Paroki Ciledug gedung gereja yang resmi selesai dibangun. Kedua daerah tersebut terletak seperti mengapit wilayah yang dinaungi Gereja Santa Bernadet Paroki Ciledug sehingga umat cenderung terbiasa mengikuti misa yang diadakan pada tempat misa yang paling dekat dengan rumahnya. Begitu pula penugasan misdinar dilakukan berdasarkan domisili mereka karena terdapat sebagian besar dari anggota misdinar Gereja Santa Bernadet Paroki Ciledug yang masih harus diantar-jemput oleh orang tuanya.

Pada Gereja Santa Bernadet Paroki Ciledug, cara pelaksanaan misa dan ibadat sebelum memasuki masa pandemi COVID-19 dan ketika masa pandemi saat ini mengalami perubahan yang menyesuaikan dengan kondisi pandemi dalam rangka mengurangi risiko terjadinya penularan. Cara pelaksanaan yang dimaksud meliputi jadwal pengadaan misa dan ibadat, dan tata cara pelayanan misdinar

dilakukan. Ketika masa pandemi, pengadaan misa setiap minggunya dikurangi. Jumlah misdinar yang dibutuhkan untuk melayani misa juga dikurangi, yaitu 1-4 orang. Dari sini, dapat diasumsikan bahwa terdapat kemungkinan cara pelaksanaan misa dan ibadat pada saat ini akan berubah menjadi seperti semula jika pandemi COVID-19 sudah berakhir nantinya.

Pada penelitian ini, cara pelaksanaan yang akan dibahas adalah cara pelaksanaan misa dan ibadat pada masa sebelum pandemi. Selain karena permasalahannya yang lebih rumit dibandingkan dengan ketika masa pandemi, solusi yang dihasilkan berdasarkan cara pelaksanaan yang seperti semula akan lebih berguna dalam jangka waktu yang panjang. Pemerintah Indonesia pada saat ini sedang mulai merencanakan penyesuaian kebijakan dalam rangka memasuki masa transisi menuju endemi dari pandemi [12]. Berdasarkan kondisi pandemi di Indonesia yang terlihat membaik tersebut, cara pelaksanaan misa dan ibadat ketika masa pandemi kemungkinan besar tidak lama lagi akan ditinggalkan.

Berikut ini adalah 2 jenis misa yang dibahas pada penelitian ini yang dibedakan berdasarkan tata cara pelayanan misdinar sebelum memasuki masa pandemi di Gereja Santa Bernadet Paroki Ciledug.

2.1.1 Misa biasa

Misa biasa yang dimaksud adalah misa yang diselenggarakan setiap pekan pada masa biasa yang meliputi Misa Hari Minggu dan Misa Sabtu Sore [13]. Tabel 2.1 di bawah ini merupakan jadwal pengadaan Misa Hari Minggu dan Misa Sabtu Sore di Gereja Santa Bernadet Paroki Ciledug.

Tabel 2.1. Jadwal pengadaan misa biasa

	Pinang	Metro Permata
Sabtu	18.30 WIB	-
Minggu	06.00 WIB	07.00 WIB
	08.30 WIB	17.00 WIB
	16.30 WIB	

sumber: wawancara dengan BPH misdinar

Jumlah misdinar yang ditugaskan untuk melayani misa biasa di Pinang dan Metro Permata secara berturut-turut adalah 12 orang dan 6 orang. Pada kasus gereja Pinang, 12 misdinar yang ditugaskan pada suatu misa harus terdiri dari 6 orang misdinar lama (senior) dan 6 orang misdinar yang baru dilantik (junior). Sedangkan pada kasus gereja Metro Permata, 6 misdinar yang ditugaskan pada suatu misa

harus terdiri dari 4 orang misdinar senior dan 2 orang misdinar junior. Komposisi misdinar yang baru dilantik dan yang sudah cukup berpengalaman tersebut juga berlaku pada jenis misa lainnya. Selain itu, misdinar yang memiliki saudara yang juga merupakan misdinar harus ditugaskan bersamaan jika memang diminta demikian. Kriteria misdinar yang bersaudara tersebut juga berlaku pada jenis misa lainnya.

2.1.2 Misa hari raya wajib

Misa hari raya wajib yang dimaksud adalah misa yang memperingati hari raya wajib dalam Gereja Katolik seperti Hari Raya Natal, Hari Raya Kenaikan Yesus, Hari Raya Santa Perawan Maria Diangkat ke Surga, Hari Raya Tubuh dan Darah Kristus, dan lain-lain. Misa untuk memperingati hari raya wajib tersebut diadakan sesuai dengan hari dan tanggal hari raya wajib yang sudah ditetapkan. Sehingga, misa hari raya wajib tersebut dapat jatuh pada hari kerja. Konferensi para uskup mungkin meniadakan atau memindahkan hari raya wajib tersebut ke hari minggu supaya umat dapat merayakannya di akhir pekan [14]. Pada penelitian ini, misa hari raya wajib yang dibahas adalah hanya misa hari raya wajib yang jatuh pada hari-hari di akhir pekan. Misa untuk memperingati hari raya wajib yang dirayakan di akhir pekan diadakan pada hari-hari yang sama dengan hari-hari ketika misa biasa diadakan seperti pada Tabel 2.1.

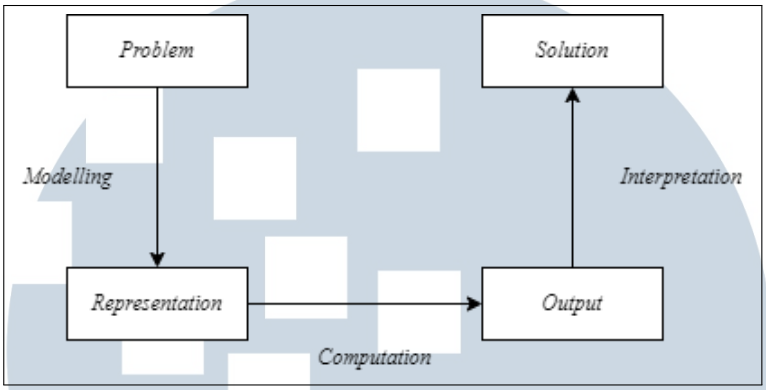
Jumlah misdinar yang ditugaskan untuk melayani misa hari raya wajib sama dengan halnya pada misa biasa. Tetapi, dari antara seluruh misdinar yang ditugaskan pada misa hari raya wajib, dibutuhkan paling sedikit 1 orang yang sudah berpengalaman menjadi pemegang wiruk atau dalam Bahasa Inggris disebut sebagai *thurifer*.

2.2 *Answer Set Programming* (ASP)

Answer Set Programming (ASP) adalah sebuah pendekatan *declarative problem solving* atau pemecahan masalah yang deklaratif yang memecahkan masalah dengan mendeskripsikan kepada komputer ”apa masalahnya” dan membiarkan komputer menemukan solusi dari masalah itu sendiri daripada mendeskripsikan kepada komputer ”bagaimana cara memecahkan masalahnya” [9].

Gambar 2.1 memperlihatkan proses dari *declarative problem solving* yang terdiri dari 3 tahap: *modelling* untuk memodelkan *problem* menjadi *representation*,

computation yang mengomputasi *representation* menjadi suatu *output*, dan *interpretation* yang menginterpretasikan *output* menjadi *solution*



Gambar 2.1. *Declarative problem solving*
sumber: [9]

.

ASP mengombinasikan *modelling language* atau bahasa pemodelan yang kaya namun sederhana dengan performa kapasitas pemecahan masalah yang tinggi. ASP berakar dari *(logic-based) knowledge representation and reasoning*, *(deductive) databases*, *constraint solving* (secara spesifik *Satisfiability Testing* (SAT)), dan *logic programming (with negation)*. ASP memungkinkan pemecahan seluruh masalah pencarian dalam NP (dan juga NP^{NP}) dengan cara yang *uniform* atau seragam. ASP mencakup di berbagai daerah penelitian seperti sistem biologis, robotik, pendukung keputusan untuk *space shuttle*, sistem desain, komposisi lagu, dan lain-lain.

2.2.1 *Logical Preliminaries* dan *Terminology*

Tabel 2.2 menjabarkan notasi yang digunakan pada berbagai macam tingkat deksripsi yang relevan dalam ASP.

Tabel 2.2. Ketentuan notasi berdasarkan tingkat deskripsi

	true, false	*if*	*and*	*or*	*iff*	*default negation*	*classical negation*
source code		:-	,	—		not	-
logic program		←	,	;		∼	¬
formula	⊤, ⊥	→	∧	*lor*	↔	∼	¬

sumber: [9]

Berikut ini adalah ketentuan dalam mendenotasikan beberapa istilah dalam ASP.

- *Predicate*, disimbolkan oleh huruf kecil seperti $a, b, ...$ untuk predikat dengan *zero arity* atau jika tidak, disimbolkan oleh huruf kecil seperti $p, q, ...$ atau *string* yang diawali dengan huruf kecil seperti *hot* atau *hasCookie*
- *Function*, disimbolkan oleh huruf kecil seperti $c, d, ...$ untuk *constant*, atau jika tidak, disimbolkan oleh huruf kecil seperti $f, g, ...$ atau *size*
- *Variable*, disimbolkan dengan huruf besar seperti X, Y, Z atau *string* yang diawali dengan huruf besar, seperti *Mother*

Arity dari *predicate* dan *function* disimbolkan dengan huruf n, dan dapat dituliskan secara eksplisit menjadi p/n untuk predikat atau f/n untuk *function*. *Term* dan *atom* didefinisikan seperti biasanya pada *atomic sentences* dalam pemrograman deklaratif.

2.2.2 *Stable Model* atau *Answer Set*

A *Informal Definition*

$$q \wedge (q \wedge \neg r \rightarrow p) \tag{2.1}$$

Misalkan α adalah sebuah *logical formula* atau proposisi logika matematika seperti dijabarkan pada (2.1) yang memiliki 3 buah model (klasik): $\{p, q\}$, $\{q, r\}$, dan $\{p, q, r\}$. Proposisi α mempunyai sebuah *stable model* yang disebut sebagai *answer set*: $\{p, q\}$. $\sqcap_\alpha$ adalah *logic program* dari proposisi α yang dijabarkan seperti berikut ini.

$$\begin{aligned} q &\leftarrow \\ p &\leftarrow q, not\, r \end{aligned} \tag{2.2}$$

Secara informal, sebuah sekumpulan atom X adalah sebuah *answer set* dari sebuah *logic program* $\sqcap$ jika:

- X merupakan sebuah model (klasik) dari $\sqcap$, dan
- semua atom pada X dibenarkan oleh sebagian *rule* pada $\sqcap$

A.1 *Syntax*

Sebuah *rule* yang dinotasikan dengan r diekspresikan dengan bentuk sebagai berikut.

$$H \leftarrow B_1, ..., B_m, \sim B_{m+1}, ..., \sim B_n. \tag{2.3}$$

Head dan *body* dari r secara berturut-turut dinotasikan dengan $head(r) = H$ dan $body(r) = \{B_1, ..., B_m, \sim B_{m+1}, ..., \sim B_n\}$. *Head* H dapat diisi dengan disjungsi $a_1 \vee ... \vee a_k$ dari atom $a_1, ..., a_k$, atau sebuah *weight constraint* dengan bentuk sebagai berikut.

$$l\#sum[\ell_1 = w_1, ..., \ell_k = w_k]u \tag{2.4}$$

$\ell_i = a_i$ atau $\ell_i = \sim a_i$ adalah sebuah literal, dan w_i adalah bilangan natural (non-negatif) yang merupakan representasi dari *weight* untuk $a_i \in$ sekumpulan huruf alfabet α dan $1 \leq i \leq k$. l dan u adalah bilangan bulat yang secara berturut-turut merupakan representasi dari *lower bound* dan *upper bound*. Jika l dan u tidak dituliskan, maka nilai l akan dianggap 0 dan nilai u dianggap ∞. Sebuah *rule* r disebut sebagai *integrity constraint* jika $head(r) = \bot$ atau H berisikan disjungsi yang kosong. Setiap komponen *body* B_i merupakan sebuah atom atau sebuah *weight constraint* untuk $1 \leq i \leq n$. Jika $body(r) = \emptyset$, r disebut sebagai *fact* atau fakta, dan biasanya tanda $\leftarrow$ dihilangkan dalam penulisan fakta. Sebuah *logic program* adalah sekumpulan *rule* yang terhingga.

A.2 *Semantics*

$$X = C_n(\sqcap^X) \tag{2.5}$$

Sekumpulan atom X adalah *stable model* atau *answer set* dari sebuah program $\sqcap$, jika persamaan 2.5 terpenuhi. $C_n(\sqcap^X)$ merupakan denotasi dari $\subseteq$-*smallest model* dari $\sqcap^X$. $\sqcap^X$ adalah *reduct* dari sebuah program $\sqcap$ yang relatif terhadap sekumpulan atom X yang mana didefinisikan sebagai berikut.

$$\sqcap^X = \{H \leftarrow B_1, ..., B_m | r \in \sqcap \text{ and } \{B_{m+1}, ..., B_n\} \cap X = \emptyset\} \tag{2.6}$$

2.2.3 *Grounding*

Sebuah *rule* yang telah melewati proses *grounding*, atau *ground rule* adalah *rule* yang tidak terdapat *variable* apa pun di dalamnya [9]. Sehingga, *term* dan *atom* di dalam *ground rule* disebut *variable-free*. Sebuah *ground instance* dari sebuah *atom* diperoleh dengan menggantikan semua *variable* yang dimiliki *atom* dengan *ground terms*. *Ground atom* diidentifikasi dengan proposisi dan juga dinotasikan dengan huruf kecil seperti $a, b, ...$

Misalkan program P seperti pada Gambar 2.2. Maka, *ground instantiation* dari program P pada Gambar 2.2 adalah $grd(P) = \bigcup_{r \in P} grd(r)$. Kumpulan $grd(r)$ yang terdiri dari *ground instances* dari sebuah *rule r* adalah kumpulan dari semua *ground rule* yang diperoleh dengan menggantikan semua *variable* di dalam *rule r* dengan *ground terms*.

```
1 arc(1,1).
2 arc(1,2).
3 edge(X,Y) :- arc(X,Y), arc(Y,X).
```

Gambar 2.2. Program P
sumber: [9]

Sehingga, *ground program* dari P atau $grd(P)$ adalah seperti yang terlihat pada Gambar 2.3. Tiga *ground rule* terakhir pada $grd(P)$ yang terlihat pada Gambar 2.3, yaitu *rule* pada *line* nomor 4,5, dan 6 jika dihilangkan akan tetap memiliki *stable model* yang sama.

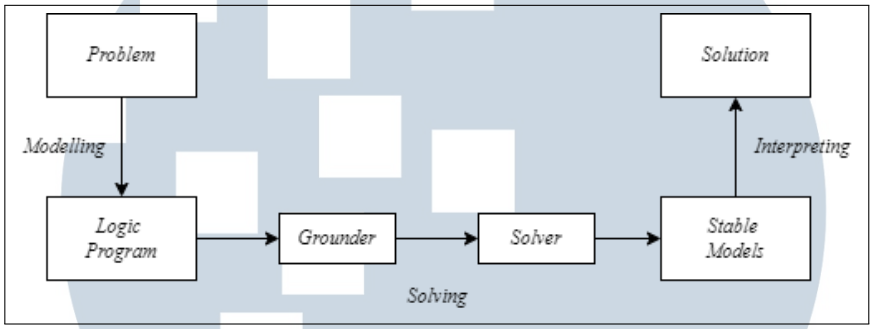
```
1 arc(1,1).
2 arc(1,2).
3 edge(1,1) :- arc(1,1), arc(1,1).
4 edge(2,2) :- arc(2,2), arc(2,2).
5 edge(1,2) :- arc(1,2), arc(2,1).
6 edge(2,1) :- arc(2,1), arc(1,2).
```

Gambar 2.3. *Ground* program P
sumber: [9]

2.2.4 *ASP Solving Process*

Secara keseluruhan, proses pemecahan masalah dengan ASP atau *ASP solving process* dapat disimpulkan dengan diagram pada Gambar 2.4. Pertama-

tama, sebuah masalah dimodelkan dengan menggunakan *syntax* dari *(first-order) logic program*. Setelah itu, dilanjutkan dengan *grounding* yang menghasilkan sebuah *finite propositional representation* dari *input program*. Lalu, sebuah solver melakukan komputasi terhadap *stable model* dari *propositional program*. Terakhir, solusinya dibacakan oleh *stable model* yang dihasilkan.



Gambar 2.4. *ASP solving process*
sumber: [9]

2.2.5 Contoh *Logic Program*

Misalkan ada sekumpulan makalah penelitian yang setiap makalahnya perlu dinilai oleh 3 orang penguji. Pengalokasian penguji terhadap makalah yang akan diujinya diharapkan sesuai dengan preferensi masing-masing penguji. Setiap penguji memiliki 2 pilihan makalah penelitian yang paling diminati: makalah pilihan pertama dan kedua. Makalah pilihan pertama lebih diminati dibandingkan makalah pilihan kedua. Setiap penguji juga memiliki 1 pilihan makalah penelitian yang paling tidak diminati (*conflict of interest*). Seorang penguji tidak boleh dialokasikan untuk menguji makalah penelitian yang paling tidak diminatinya.

Gambar 2.5 adalah *problem instance* dari permasalahan pengalokasian penguji makalah penelitian berdasarkan penjelasan di atas.

```
paper(p1).
paper(p2).
paper(p3).
paper(p4).
[...]

reviewer(r1). classA(r1,p1). classB(r1,p2). coi(r1,p3).
reviewer(r2). classA(r2,p3). classB(r2,p4). coi(r2,p6).
[...]
```

Gambar 2.5. Contoh *problem instance*
sumber: [9]

Gambar 2.6 adalah *problem encoding* dari permasalahan pengalokasian penguji makalah penelitian berdasarkan penjelasan yang terdapat pada penjelasan permasalahan yang telah dijabarkan.

```
{ assigned(P,R) : reviewer(R) } = 3 :- paper(P).

:- assigned(P,R), coi(R,P).
:- assigned(P,R), not classA(R,P), not classB(R,P).
:- not 6 { assigned(P,R) : paper(P) } 9, reviewer(R).

assignedB(P,R) :- classB(R,P), assigned(P,R).
:- 3 { assigned(P,R) : paper(P) }, reviewer(R).

#minimize { 1,P,R : assignedB(P,R), paper(P), reviewer(R) }.
```

Gambar 2.6. Contoh *problem encoding*
sumber: [9]