Hak cipta dan penggunaan kembali:

Lisensi ini mengizinkan setiap orang untuk menggubah, memperbaiki, dan membuat ciptaan turunan bukan untuk kepentingan komersial, selama anda mencantumkan nama penulis dan melisensikan ciptaan turunan dengan syarat yang serupa dengan ciptaan asli.

Copyright and reuse:

This license lets you remix, tweak, and build upon work non-commercially, as long as you credit the origin creator and license it on your new creations under the identical terms.

Team project ©2017
Dony Pratidana S. Hum | Bima Agus Setyawan S. IIP

BAB II

LANDASAN TEORI

2.1 Kajian Penjadwalan Kegiatan

Penjadwalan berasal dari kata "jadwal", yang berarti sebuah rencana nyata atau dokumen berisi kapan seharusnya sebuah kejadian akan terjadi (Baker, 2009). Sebuah jadwal biasanya memberi informasi waktu yang direncanakan untuk terlaksananya sebuah tugas dan menjawab pertanyaan, "Bila semua berjalan lancar, maka kapan sebuah kegiatan akan terjadi?".

Dalam jurnal "Principles of Sequencing and Scheduling", Baker (2009) berpendapat bahwa pertanyaan 'kapan' tersebut dapat dijawab dalam dua bentuk: waktu dan bentuk urutan kegiatan. Dalam bentuk waktu, misalnya ujian akan dilaksanakan pada pukul 09.00 WIB atau makan siang pada pukul 12.30 WIB. Dalam bentuk urutan kegiatan, misalnya bus akan berangkat setelah semua penumpang naik atau kelas akan dimulai setelah guru datang. Kedua bentuk waktu dan urutan kegiatan inilah yang menjadi elemen utama dalam pembuatan sebuah jadwal. Berhubungan dengan makna tersebut, pembuatan sebuah jadwal menjadi definisi dari sebuah penjadwalan. Dalam proses ini, ada dua pendekatan yang dapat diambil yaitu pengurutan atau penjadwalan. Dalam pengurutan, yang direncanakan adalah urutan dari sebuah kegiatan atau pengambilan keputusan untuk memilih kegiatan berikutnya. Dalam

Penerapan Algoritma ..., Putriasihta Baeha, FTI UMN, 2015

penjadwalan, yang direncanakan adalah waktu kapan kegiatan tersebut akan dimulai dan mungkin kapan kegiatan tersebut akan selesai.

Meskipun demikian, dalam kehidupan nyata ada banyak kejadian tak terduga yang mungkin dapat terjadi. Oleh karena itu Baker (2009) juga berpendapat bahwa perlu dipikirkan *buffer time* atau waktu cadangan agar jadwal yang dibuat masih bisa diikuti oleh para pemakainya dengan aman. Misalnya bila bus dijadwalkan akan berangkat pada 08.15 WIB, sebenarnya supir bus ditargetkan untuk sudah siap pada pukul 08.00 WIB. Dengan demikian, ada *buffer time* 15 menit untuk memperhitungkan waktu macet di jalan dan kejadian tak terduga lainnya.

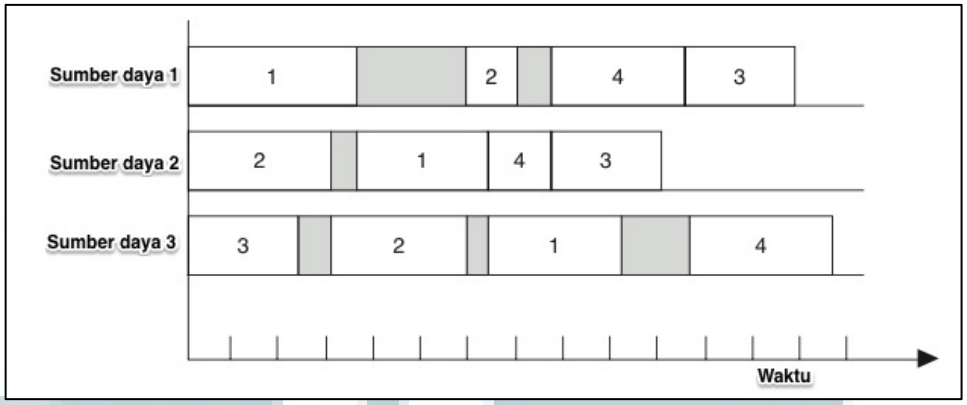
Baker (2009) melanjutkan, struktur utama dari sebuah penjadwalan adalah tugas yang perlu dilaksanakan dan sumber daya yang tersedia untuk mengerjakan tugas tersebut.

Masalah penjadwalan dalam industri juga memiliki struktur yang sama. Terdapat sejumlah tugas yang harus dilaksanakan dan sejumlah sumber daya yang dibutuhkan untuk menyelesaikan tugas tersebut. Dengan tugas dan sumber daya yang ada bersama dengan informasi tentang hal tak terduga yang mungkin terjadi, penjadwalan dilakukan untuk menentukan kapan sebuah tugas harus selesai dengan memperhitungkan kemampuan sumber daya yang ada untuk menyelesaikan tugas tersebut (Baker, 2009). Semua ini didefinisikan dalam sebuah bentuk formal agar dapat dimengerti dengan lebih mudah.

Bentuk formal yang umum dipakai adalah Gantt Chart pada gambar 2.1. Diagram ini menunjukkan sumber daya yang dialokasikan untuk waktu tertentu. Sumber daya diletakkan pada sumbu Y dan waktu yang dibutuhkan diletakkan pada

Penerapan Algoritma ..., Putriasihta Baeha, FTI UMN, 2015

sumbu X. Dengan demikian, diagram tersebut akan menunjukkan waktu yang dibutuhkan sumber daya tertentu untuk menyelesaikan sebuah kegiatan.



Gambar 2.1 Ilustrasi Gantt Chart sederhana (Baker, 2009)

Dalam pembuatan sebuah jadwal, ada syarat-syarat yang harus dipenuhi sebelum aktivitas yang dijadwalkan dapat terlaksana. Sebuah jadwal disebut "layak" apabila seluruh kegiatan dapat diselesaikan menurut syarat atau batasan yang ada (Buttazzo, 1997). Kegiatan yang akan dilaksanakan disebut "dapat dijadwalkan" apabila paling tidak ada satu algoritma yang dapat dipakai untuk menghasilkan jadwal yang layak.

Syarat-syarat yang harus dipenuhi agar sebuah kegiatan dalam jadwal dapat terlaksana disebut *constraints*. *Constraints* dapat dikategorikan menjadi tiga tipe (Buttazzo, 1997):

a. *Timing constraint* adalah syarat waktu yang harus dipenuhi sebuah kegiatan. Istilah yang sering digunakan adalah *deadline*, yang berarti kegiatan tersebut harus diselesaikan tanpa masalah sebelum atau pada waktu yang telah ditentukan. Misalnya, makan siang harus sudah disiapkan pada jam 12.30 WIB.

Penerapan Algoritma ..., Putriasihta Baeha, FTI UMN, 2015

b. *Precedence constraint* adalah syarat berdasarkan urutan yang harus dipenuhi sebuah kegiatan. Hal ini berarti kegiatan B hanya bisa dilaksanakan setelah kegiatan A selesai dan pelaksanaannya bergantung pada hasil kegiatan A (apakah kegiatan itu gagal atau berhasil).

   Misalnya, untuk memasak sebuah sup bakso, pertama harus membeli bakso terlebih dahulu.

c. *Resource constraint* adalah syarat berdasarkan kualitas (bobot atau kemampuan) dan kuantitas (jumlah) dari sumber daya yang dapat mengerjakan sebuah kegiatan. Sumber daya dapat berupa banyak hal seperti manusia, komputer, sebuah program, dan lain sebagainya.

   Sebuah sumber daya yang hanya akan mengerjakan sebuah tugas tertentu disebut *private resource*, sedangkan sumber daya yang dapat digunakan oleh lebih dari satu kegiatan disebut *shared resource*.

   Misalnya, sumber daya yang diperlukan untuk memasak semangkok bakso adalah panci dan kompor gas.

Ketika *constraint* tersebut diaplikasikan dalam sebuah algoritma, tiga *constraint* tersebut bisa mendapat dua bentuk perlakukan yaitu diperlakukan sebagai *hard constraint* (syarat yang harus dipenuhi, apapun yang terjadi) dan *soft constraint* (diupayakan untuk dipenuhi).

Dengan memperhatikan faktor-faktor di ataslah sebuah penjadwalan bisa dibuat dengan efektif.

Penerapan Algoritma ..., Putriasihta Baeha, FTI UMN, 2015

2.2 Algoritma Scatter Search

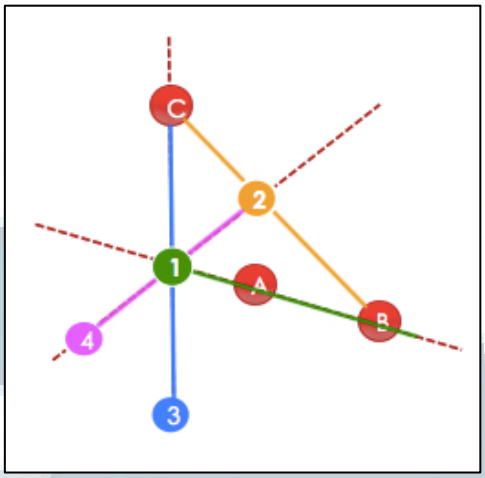
Scatter search yang secara harafiah berarti "pencarian menyebar" pertama kali diperkenalkan oleh Fred Glover pada tahun 1977 sebagai sebuah *heuristic* dalam pemrograman. Algoritma ini adalah hasil pengembangan dari *Tabu Search*.

Scatter search adalah algoritma *metaheuristic* dan optimisasi global yang bekerja dengan mengkombinasikan berbagai macam kemungkinan solusi yang ada dan berbeda dengan menghitung bobot dan *constraint* dari tiap solusi. Algoritma ini menghasilkan solusi yang berbeda dan variatif secara sistematik. Tujuannya adalah menghasilkan sebuah prosedur solusi yang lebih baik dibandingkan dengan proposal sistem asli yang hanya menghasilkan solusi dari satu karakteristik saja (Marti, 2003).

Algoritma ini mengeksplorasi kemungkinan solusi yang ada secara sistematis dengan menggunakan *set of reference* yang berisi solusi-solusi 'baik' – yang didapatkan dari pemecahan masalah sebelumnya. Kriteria 'solusi baik' ini tidak hanya berdasarkan pada fungsi objektif. Selain itu, kriteria 'solusi baik' tersebut tidak terbatas hanya pada satu solusi, tetapi bisa diaplikasikan terhadap beberapa solusi sesuai dengan kebutuhan.

Menurut Glover (2003), cara kerja *Scatter search* dapat diilustrasikan melalui gambar 2.2.

Penerapan Algoritma ..., Putriasihta Baeha, FTI UMN, 2015



Gambar 2.2 Ilustrasi *Scatter Search* (Glover, 2003)

Dalam ilustrasi tersebut A, B, dan C adalah solusi awal yang dimiliki oleh *set of reference* awal. Solusi baru dihasilkan dari kombinasi linear oleh paling tidak dua solusi referensi. Solusi 1 didapatkan dari kombinasi solusi referensi A dan B. Sebenarnya ada banyak solusi yang dihasilkan dari kombinasi tersebut, tapi hanya solusi 1 yang dimasukkan ke dalam *set of reference* solusi berdasarkan kriteria optimal.

Dengan cara kerja yang sama, kombinasi solusi B dan C menghasilkan solusi 2; kombinasi solusi C dan 1 menghasilkan solusi 3; dan kombinasi solusi 1 dan 3 menghasilkan solusi 4. Dengan demikian, *set of reference* solusi baru bisa didapatkan dengan sistematis.

2.2.1 Implementasi Umum Algoritma *Scatter Search*

Metodologi *scatter search* sangat fleksibel karena setiap elemennya dapat diimplementasikan dalam berbagai cara. Glover (2003) menyatakan berikut adalah lima tahapan dasar implementasi algoritma *Scatter Search*.

Penerapan Algoritma ..., Putriasihta Baeha, FTI UMN, 2015

a. *Diversification Generation Method* (Tahap Generasi Diversifikasi) adalah tahapan pengumpulan beberapa solusi percobaan yang berbeda dan variatif, dengan menggunakan solusi sementara (*seed solution*) sebagai *input*.

   Set solusi sementara didapatkan dengan *random*.

b. *Improvement Method* (Tahap Perbaikan) adalah tahapan untuk mengubah solusi percobaan menjadi lebih baik. Solusi *input* atau *output* tidak diharuskan untuk menjadi solusi optimal, tetapi biasanya solusi output diharapkan untuk lebih baik dari input. Bila tidak ada perbaikan, maka solusi yang telah 'diperbaiki' dianggap sama dengan solusi *input*.

   Standar 'perbaikan' didapat dengan mencocokkan solusi percobaan dengan *constraint-constraint* yang ada.

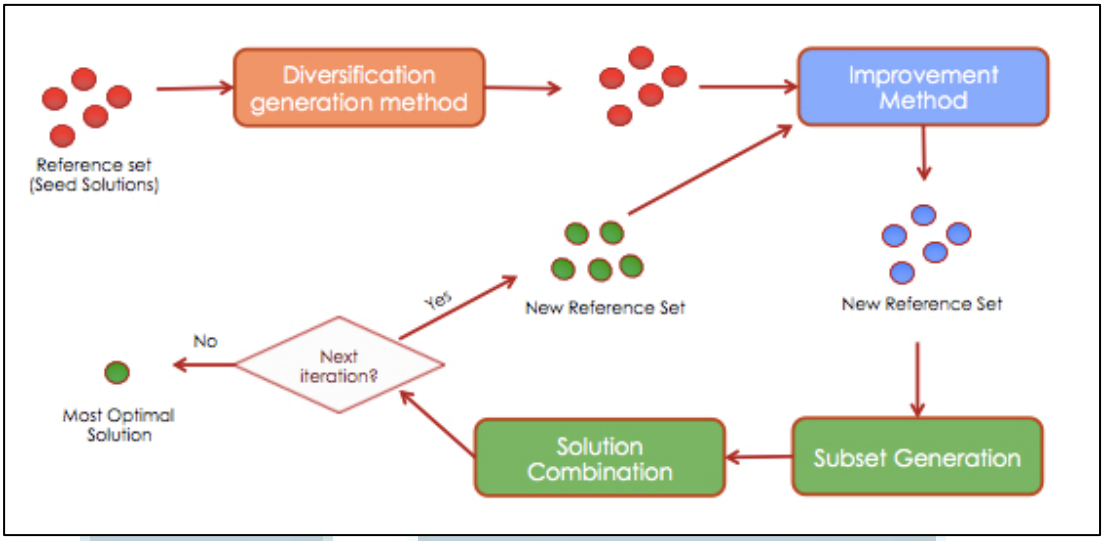
c. *Reference Set Update Method* (Tahap *Update* Set Referensi) adalah tahapan untuk membangun sebuah *reference set* yang berisi solusi "terbaik" berjumlah *b* (biasanya berjumlah kecil, tidak lebih dari 20). Solusi percobaan dapat masuk ke dalam *reference set* berdasarkan kualitas bobot atau diversitasnya dari solusi yang lain.

d. *Subset Generation Method* (Tahap Generasi *Subset*) adalah tahap mengerjakan *reference set* untuk menghasilkan sebuah *subset* solusi yang menjadi dasar untuk membuat solusi kombinasi.

e. *Solution Combination Method* (Tahap Kombinasi Solusi) adalah tahap untuk mengubah *subset* solusi yang dihasilkan oleh *Subset Generation Method* untuk menjadi satu atau lebih vektor kombinasi solusi.

Penerapan Algoritma ..., Putriasihta Baeha, FTI UMN, 2015

Secara sederhana, lima tahapan *Scatter Search* tersebut dapat digambarkan oleh gambar 2.3.



Gambar 2.3 *Flowchart* Scatter Search (Marti, 2003)

Kumpulan *reference set* awalnya berisi *seed solutions* yang kosong, lalu kemudian akan diisi secara *random* dengan menggunakan *Diversification Generation Method*. Calon solusi tersebut kemudian akan mengalami *Improvement Method*, di mana kumpulan solusi tersebut akan mengalami tahap perbaikan untuk menghasilkan *reference set* yang lebih baik.

Reference set tersebut lalu akan disatukan dengan *Reference Set Update Method.* Kemudian *Subset Generation Method* akan membuat kumpulan solusi untuk dicocokkan dengan *reference set.*

Pada akhirnya, *Solution Combination Method* akan mencocokkan solusi dengan *reference set* untuk menghasilkan solusi yang paling optimal.

Proses dari lima tahapan tersebut dapat dijabarkan dalam gambar 2.4.

```
Input: DiverseSet_size, ReferenceSet_size
Output: ReferenceSet
InitialSet ← ConstructInitialSolution(DiverseSet_size);
RefinedSet ← ∅;
for S_i ∈ InitialSet do
    RefinedSet ← LocalSearch(S_i);
end
ReferenceSet ← SelectInitialReferenceSet(ReferenceSet_size);
while ¬ StopCondition() do
    Subsets ← SelectSubset(ReferenceSet);
    CandidateSet ← ∅;
    for Subset_i ∈ Subsets do
        RecombinedCandidates ← RecombineMembers(Subset_i);
        for S_i ∈ RecombinedCandidates do
            CandidateSet ← LocalSearch(S_i);
        end
    end
    ReferenceSet ← Select(ReferenceSet, CandidateSet,
    ReferenceSet_size);
end
return ReferenceSet;
```

Gambar 2.4 *Pseudocode* Scatter Search (Glover, 2003)

Pseudocode di atas dapat dijelaskan sebagai berikut.

1. Mulai dengan $P = \emptyset$. Gunakan *Diversification Generation Method* untuk membangun satu solusi, kemudian gunakan *Improvement Method* terhadap solusi tersebut. Anggap solusi tersebut adalah x. Bila $x \notin P$ maka tambahkan x ke dalam P (misal, $P = P \cup x$), bila tidak, hapus x. Ulangi langkah ini sampai $|P| = PSize$ (memenuhi ukuran yang diinginkan).
2. Gunakan *Reference Set Update* untuk membangun set referensi solusi *RefSet* $= \{ x^1, ..., x^b \}$ dengan "solusi terbaik" b di dalam P. Urutkan solusi-solusi di dalam *RefSet* menurut bobot objektif fungsinya sehingga didapatkan x^1 sebagai solusi terbaik dan x^b sebagai yang terburuk.

Buat *NewSolutions* = TRUE.

while (*NewSolutions*) **do**

3. Bentuk *NewSubsets* dengan *subset generation method*. Buat *NewSolutions* = FALSE.

while (*NewSubsets* $\neq \emptyset$) **do**

4. Pilih subset *s* berikutnya di dalam *NewSubsets*.
5. Gunakan *solution combination method* pada *s* untuk mendapatkan satu atau lebih calon solusi *x*. Gunakan improvement method ke calon solusi tersebut.
6. Gunakan fungsi *reference set update*.

if (*RefSet* has changed) **then**

7. Jadikan *NewSolutions* = TRUE.

end if

8. Hapus *s* from *NewSubsets*.

end while

end while

2.2.2 Contoh Implementasi Algoritma *Scatter Search* dalam Sebuah Sistem Penjadwalan

Burke (2007) mengajukan implementasi untuk model penjadwalan *shift* pekerjaan suster di rumah sakit dengan menggunakan algoritma *Scatter Search*. Berikut adalah penjabaran penggunaan lima tahapan dasar algoritma tersebut dalam sebuah sistem penjadwalan.

Penerapan Algoritma ..., Putriasihta Baeha, FTI UMN, 2015

A. *Diversification Generation Method*

Metode ini digunakan untuk membuat set solusi yang bervariasi. Solusi ini kemudian akan diperbaiki (menurut fungsi objektif yang dibutuhkan) dan ditambahkan ke dalam set yang akan dijadikan referensi. Saat membuat set solusi yang bervariasi ini, fungsi objektif dari tiap solusi diabaikan.

Contoh *pseudocode* dari Burke (2007) untuk mendapatkan hasil tersebut dalam kasus penjadwalan *shift* suster dijelaskan melalui gambar 2.5.

```
1. Create an empty set (set) of size n for the diverse solutions
2. UNTIL set is full
3.    Create a roster (roster) with no assignments made
4.    FOR each day (day) in roster
5.       FOR each shift type (shift) to be covered on day
6.          UNTIL the cover is satisfied
7.             Assign shift to a nurse who has been assigned shift on day the
               least number of times in all other rosters in set (subject to no hard
               constraint violations)
8.             If more than one nurse has received shift on day the least number
               of times then randomly select one of them
9.          ENDUNTIL
10.      ENDFOR
11.   ENDFOR
12.   Add roster to set
13. ENDUNTIL
```

Gambar 2.5 *Pseudocode Diversification Generation Method* (Burke, 2007)

Berikut adalah penjelasan *pseudocode* tersebut.

1. Buat set kosong (*set*) dengan ukuran n untuk solusi yang terdiversifikasi
2. UNTIL *set* tersebut penuh
3. Buat jadwal (*roster*) yang kosong, tanpa tugas
4. FOR setiap hari (*day*) yang ada di dalam *roster*

Penerapan Algoritma ..., Putriasihta Baeha, FTI UMN, 2015

5. FOR setiap shift pekerjaan (*shift*) yang harus dilakukan dalam satu *day*
6. UNTIL seluruh pekerjaan terisi
7. Masukkan *shift* untuk suster yang sudah mendapatkan *shift* di dalam *day* yang paling sedikit di dalam *set* yang ada (ini adalah peraturan *hard constraint* yang tidak boleh dilanggar)
8. Bila ada lebih dari satu suster yang mendapatkan *shift* dalam *day* paling sedikit, pilih salah satu secara random
9. END UNTIL
10. END FOR
11. END FOR
12. Tambahkan *roster* ke dalam *set*
13. END UNTIL

Dengan metode ini, maka bisa didapatkan hasil set awal yang paling bervariasi.

B. *Improvement Method*

Improvement Method dieksekusi setelah solusi baru didapat dengan menggunakan *Diversification Generation Method* dan *Solution Combination Method* sebelumnya. Metode ini menerima input solusi baik yang mungkin dilakukan maupun yang mungkin. Tetapi, hasil *output*-nya haruslah solusi yang mungkin bisa dilakukan.

Menurut Laguna dan Marti (2003), sebenarnya metode ini tidak harus dilakukan dalam *Scatter Search*. *Improvement Method* hanya perlu untuk diimplementasikan apabila program harus mengolah data yang sangat banyak sehingga jumlah set calon solusi yang sangat bervariasi dan bertujuan untuk

Penerapan Algoritma ..., Putriasihta Baeha, FTI UMN, 2015

mendapatkan hasil dengan kualitas sangat tinggi, walaupun harus mengorbankan kecepatan komputasi.

Meskipun demikian, contoh paling sederhana dalam mengaplikasikan *improvement method* ini adalah dengan mencocokkan set calon solusi dengan *soft constraint* yang ada sebanyak mungkin.

C. *Reference Set Update Method*

Reference Set Update Method digunakan dalam dua segmen yang berbeda di algoritma ini. Metode ini digunakan untuk membuat set referensi awal dari calson solusi yang telah dibuat oleh *Diversification Method*. Setelah itu, metode ini digunakan untuk menjaga set referensi. Metode ini nantinya akan menentukan apakah solusi yang dihasilkan oleh metode kombinasi dan *improvement* akan ditambahkan ke set referensi atau tidak. Karena nantinya, set referensi tersebutlah yang akan menjadi hasil akhir solusi paling optimal.

Untuk metode ini, Burke menggunakan cara yang juga digunakan oleh Glover (2003). Setelah menerima input solusi yang diberikan oleh *Diversification Method* dan dioptimalkan oleh *Improvement Method*, solusi tersebut diberi peringkat berdasarkan fungsi objektifnya. Atau dengan kata lain, yang paling memenuhi bobot *constraint* yang ada.

Solusi terbaik yang paling memenuhi (*b1*) kemudian ditambahkan ke dalam set referensi. Dari solusi yang tersisa, *b2* dipilih dan ditambahkan ke dalam set referensi – berdasarkan kontribusi perbedaan / keragaman yang bisa diberikan kepada set referensi.

Penerapan Algoritma ..., Putriasihta Baeha, FTI UMN, 2015

Bila makin beragam, maka kemungkinan untuk mendapatkan solusi ideal pun semakin besar. Gambar 2.6. menampilkan *pseudocode* dari Burke untuk metode ini.

P is the set of solutions created using the diversification generation method.
RefSet is the reference set and is initially empty.
b_1 and b_2 are algorithm specific parameters (integers >= 0).

1. FOR 1 to b_1
2. Select from *P* the best solution according to the objective function
3. Remove the solution from *P* and add it to *RefSet*
4. ENDFOR
5. FOR 1 to b_2
6. For each solution in *P* calculate its total similarity to all the solutions currently in *RefSet* (using the similarity function)
7. Select the least similar solution (the schedule with least assignments in common with other rosters in *RefSet*)
8. Remove the solution from *P* and add to *RefSet*
9. ENDFOR

Gambar 2.6 *Pseudocode Reference Set Update Method* (Burke, 2007)

Berikut penjelasan *pseudocode* di gambar 2.6 atas.

Anggap *P* adalah set solusi yang telah dihasilkan oleh *Diversification Generation Method.*

RefSet adalah set referensi dan pada awalnya kosong.

b1 dan *b2* adalah parameter spesifik untuk algoritma ini (berupa bilangan integral >=0).

1. FOR 1 to *b1*
2. SELECT solusi terbaik dari P menurut fungsi objektif
3. Hapus solusi tersebut dari P dan tambahkan ke dalam *RefSet*
4. ENDFOR
5. FOR 1 to *b2*

Penerapan Algoritma ..., Putriasihta Baeha, FTI UMN, 2015

6. Untuk setiap solusi di P, hitung kesamaannya dengan solusi yang sekarang ada di dalam *RefSet* (menggunakan fungsi perbandingan kesamaan)
7. Pilih solusi yang paling tidak sama (jadwal dengan penugasan yang paling tidak mirip dengan penugasan lain di *RefSet*)
8. Hapus solusi tersebut dari P dan tambahkan ke dalam *RefSet*
9. ENDFOR

Setelah *Solution Combination Method*, solusi baru ditambahkan ke dalam set referensi apabila nilai fungsi objektif solusi tersebut lebih baik daripada solusi 'terburuk' yang ada di dalam set referensi sekarang dan set tersebut tidak memiliki solusi yang identik.

Bila solusi baru ditambahkan, solusi terburuk yang ada di set referensi saat ini harus dihapus.

D. *Subset Generation Method*

Subset Generation Method digunakan untuk mengenali *subse*t solusi dari set referensi yang akan digunakan oleh *Solution Combination Method* untuk membuat solusi baru. Menurut Glover (2003), ada 4 tipe *subset* berbeda yang akan dibuat, yaitu:

a. Tipe 1: semua subset berisi solusi unik dari set referensi yang berisi 2 elemen.
b. Tipe 2: subset berisi 3 elemen yang didapat dengan menambahkan 'solusi terbaik' ke setiap 2 elemen subset di atas, tapi belum ada di subset ini.
c. Tipe 3: subset berisi 4 elemen yang didapat dengan menambahkan 'solusi terbaik' ke setiap 3 elemen subset di atas, tapi belum ada di subset ini.
d. Tipe 4: subset berisi solusi *i* yang terbaik, dengan i = 5 dari *RefSet*.

Penerapan Algoritma ..., Putriasihta Baeha, FTI UMN, 2015

Saat implementasi algoritma, *subset* yang dibuat menggunakan salah satu dari keempat tipe di atas.

E. *Solution Combination Method*

Metode ini menggunakan dua atau lebih solusi yang telah dipilih oleh *Subset Generation Method* sebagai referensi, untuk kemudian menghasilkan satu atau lebih solusi baru yang optimal.

Menurut Glover (2003), metode untuk mengombinasikan solusi ini spesifik untuk setiap masalah karena bergantung terhadap representasi solusi yang diinginkan.

Pada kasus penjadwalan suster yang dikerjakan oleh Burke, solusi yang dimaksud bisa berupa jumlah penugasan *shift* untuk suster yang tersedia. Dalam *Solution Combination Method*, setiap pilihan solusi yang ada dianggap sebagai sebuah *voting* untuk *kandidat*. Kandidat adalah semua kemungkinan penugasan *shift* untuk suster. Voting didapat dari kecocokan antara pilihan solusi tersebut dengan *guiding solutions*. Setiap voting yang masuk kemudian dihitung untuk membentuk solusi baru.

Dalam contoh *pseudocode* di gambar 2.7, kandidat adalah penugasan *shift* untuk suster di satu hari tertentu. Langkah pertama dalam proses ini adalah dengan membuat solusi baru yang belum memiliki isi.

Penerapan Algoritma ..., Putriasihta Baeha, FTI UMN, 2015

```
1. Identify candidates as the set of all possible shift to nurse assignments for
   this instance
2. Collect all the candidates' votes from each solution in the guiding set
3. Remove from candidates any candidate with zero votes
4. IF candidates is empty
      GOTO 10.
5. Sort candidates by:
      a) decreasing total number of votes
      b) increasing total number of votes successful for voters selecting this candidate
      c) increasing sum of objective function values for voters selecting this candidate
6. Select and remove the first candidate in candidates
7. Make the assignment represented by this candidate in the new solution
   unless it exceeds cover requirements or breaks any hard constraints
8. IF the assignment was made
      GOTO 4.
9. IF candidates is not empty
      GOTO 6.
10. Return new solution
```

Gambar 2.7 *Pseudocode Solution Combination Method (*Burke, 2007).

1. Identifikasi *kandidat* sebagai set kemungkinan penugasan *shift* untuk suster
2. Kumpulkan voting para kandidat dari setiap solusi yang ada di *guiding set solution*
3. Hapus semua kandidat yang memiliki vote 0 dari set *kandidat*
4. IF *kandidat* kosong GOTO 10.
5. Urutkan *kandidat* berdasarkan:

a. Jumlah voting dari yang paling besar ke kecil

b. Jumlah voting berhasil dari solusi yang memilih kandidat ini, dari yang paling kecil ke besar

c. Jumlah bobot fungsi objektif dari solusi yang memilih kandidat ini, dari yang paling kecil ke besar

6. Pilih kandidat pertama dalam set *kandidat* dan hapus dari set
7. Buat penugasan dengan kandidat yang telah dipilih di set solusi baru, kecuali kandidat tersebut melewati kebutuhan yang ada atau tidak memenuhi *hard constraint*

Penerapan Algoritma ..., Putriasihta Baeha, FTI UMN, 2015

8. IF penugasan sudah dilaksanakan GOTO 4.

9. IF *kandidat* tidak kosong GOTO 6.

10. RETURN solusi baru.