Penyelesaian {0,1}-*Knapsack Problem* dengan Algoritma *Soccer League Competition*

Muji Prasetyo Iryanto, Sri Mardiyati

Departemen Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Indonesia, Depok, Jawa Barat
mujipi@sci.ui.ac.id

Abstrak

Knapsack Problem (KP) adalah masalah penempatan *item* (barang) ke dalam suatu tempat (biasa disebut *Knapsack*) yang mempunyai kapasitas tertentu, dimana setiap *item* memiliki berat dan nilai, sehingga total berat dari *item-item* yang ditempatkan tidak melebihi kapasitas *Knapsack* dan nilai yang didapatkan maksimum. {0,1}-*Knapsack Problem* ({0,1}-KP) adalah kasus khusus dari KP dimana setiap *item* hanya tersedia 1 unit, sehingga keputusannya adalah untuk memasukkan *item* tersebut ke dalam *Knapsack* (x=1) atau tidak (x=0). Telah banyak metode yang dikembangkan untuk menyelesaikan {0,1}-KP, salah satunya adalah Algoritma *Soccer League Competition* (SLC). SLC adalah algoritma *meta-heuristic* yang terinspirasi dari kompetisi pada liga sepakbola. Kompetisi antar tim untuk menjadi juara dan kompetisi internal antar pemain untuk memperoleh kesempatan bermain disimulasikan untuk mencari nilai optimum dari {0,1}-KP. Hasil simulasi pada beberapa permasalahan {0,1}-KP yang telah menjadi *benchmark* menunjukkan bahwa SLC dapat digunakan untuk menyelesaikan {0,1}-KP, yang mana melampaui hasil dari algoritma lain.

Kata Kunci: *Knapsack Problem* (KP); {0,1}-*Knapsack Problem* ({0,1}-KP); Algoritma *Soccer League Competition* (SLC)

PENDAHULUAN

Knapsack Problem (KP) adalah masalah optimasi dalam penempatan N *item* (barang) ke dalam tempat (biasa disebut *Knapsack*) yang mempunyai kapasitas c, dimana setiap item memiliki berat w_j dan nilai v_j, sehingga total berat dari item-item yang ditempatkan tidak melebihi kapasitas *Knapsack*, dan jumlah nilai yang didapatkan maksimum (Kellerer *et al*, 2004). Saat jumlah setiap *item* hanya terdapat satu, maka KP akan disebut {0,1}-KP.

Telah banyak metode yang dikembangkan untuk menyelesaikan KP diantaranya *Ant Colony Optimization Algorithm* yang dikembangkan oleh Shi (2006), *Schema-Guiding Meta-Heuristi Algorithm* (SGEA) yang diperkenalkan oleh Liu dan Liu (2009), *Genetic Algorithm* yang diaplikasikan oleh Lin (2008), *Binary Particle Swarm Optimization based on Multi-Mutation Strategy* (MMBPSO) yang diperkenalkan Li (2009) untuk menyelesaikan KP, serta *Novel Global Harmony Search* (NGHS) yang dikembangkan dari *Harmony Search* oleh Zou *et al* (2011).

Hingga saat ini masih banyak pengembangan metode untuk menyelesaikan {0,1}-KP. Hal ini dikarenakan masalah {0,1}-KP masih berkembang dalam tingkat kesulitan dan dimensinya. Selain itu, beberapa metode dapat menyelesaikan {0,1}-KP berdimensi kecil dengan cepat, tetapi tidak mampu untuk menyelesaikan {0,1}-KP yang berdimensi lebih besar.

Algoritma *Soccer League Competition* (SLC) adalah algoritma baru untuk menyelesaikan sistem persamaan nonlinear (Moosavian, 2015). Kemudian, Moosavian (2015) mengaplikasikan algoritma SLC untuk mencari desain optimal dari sistem distribusi air. Ide dari algoritma SLC terinspirasi dari liga sepak bola profesional dan

proses pencarian nilai optimalnya berdasarkan dari kompetisi yang terjadi antar tim dan antar pemain (Moosavian, 2013).

Pada sebuah musim dalam liga pertandingan sepak bola, terdapat sejumlah tim yang akan saling bertanding melawan tim lain setiap minggunya. Dalam sebuah pertandingan, tim yang menang akan mendapatkan 3 poin, 1 poin untuk seri, dan 0 poin jika kalah. Pada akhir musim, tim dengan akumulasi poin terbanyak akan menjadi juara.

Kompetisi tidak hanya terjadi di tingkat antar tim, tetapi juga pada antar pemain dalam tim yang sama. Setiap tim memiliki lebih dari 11 pemain, yang terdiri dari pemain utama dan pemain cadangan. Kompetisi antar pemain terjadi karena setiap pemain ingin memiliki kesempatan menjadi pemain utama agar dapat bermain. Setiap pemain akan meningkatkan kemampuannya agar dapat menarik perhatian pelatih supaya dapat menjadi pemain utama. 11 pemain dengan kemampuan terbaiklah yang akan dipilih menjadi pemain utama. Kompetisi internal ini secara tidak langsung juga mempengaruhi peningkatan kualitas dan kemampuan tim.

Dalam setiap tim terdapat seorang pemain kunci yang disebut Pemain Bintang (*Star Player*; SP). SP memiliki kemampuan terbaik diantara pemain lain dalam tim tersebut. Lebih lanjut, terdapat satu orang dalam sebuah liga yang disebut Pemain Bintang Super (*Super Star Player*; SSP) yang memiliki kemampuan terbaik dalam sebuah liga. Sebagai contoh, misalnya, di Liga Spanyol Cristiano Ronaldo menjadi SSP dari liga dan SP dari timnya, Real Madrid. Selain itu, Lionel Messi adalah SP dari Barcelona.

Pada algoritma SLC, para pemain dari tim yang menang akan melakukan strategi peningkatan kemampuan dalam setiap akhir pertandingan. Para pemain tersebut akan mencoba mengimitasi SP dari timnya dan SSP dari liga, sehingga kemampuan mereka dapat ikut meningkat. Proses imitasi tersebut akan disimulasikan dalam Operator Imitasi dan Operator Provokasi.

SLC akan disimulasikan pada 18 masalah {0,1}-KP yang telah menjadi *benchmark*. Hasilnya akan dapat dibandingkan dengan algoritma yang telah ada sebelumnya yaitu algoritma NGHS.

PENYELESAIAN {0,1}-KP DENGAN ALGORITMA SLC

Populasi terdiri dari dua jenis: (1) tim dan (2) pemain. Kompetisi yang terjadi antar tim untuk menang dan antar pemain untuk dapat menjadi SP atau SSP dapat disimulasikan untuk masalah optimasi.

Dalam masalah optimasi, setiap pemain diibaratkan vektor solusi yang mencari posisi global optimum. Oleh sebab itu, setiap pemain di sebuah liga, pemain terbaik dalam tim, dan pemain terbaik di liga dapat diasumsikan sebagai vektor solusi, nilai optimum lokal, dan nilai optimum global.

Pencarian nilai optimum global dalam algoritma SLC dilakukan dalam beberapa langkah prosedur sebagai berikut.

Inisialisasi Masalah

Pertama-tama, masalah yang ada dinyatakan ke dalam bentuk {0,1}-KP, yaitu :
fungsi tujuan

$$maks \ \sum_{j=1}^{N} p_j x_j \qquad (1)$$

dengan kendala

$$\sum_{j=1}^{N} w_j x_j \leq c, \qquad (2)$$

$$x_j \in \{0,1\} \qquad (3)$$

dengan

p_j : nilai dari *item j*

w_j : berat dari *item j*

c : kapasitas *Knapsack*

x_j : nilai *variable* keputusan, dimana nilai 1 berarti item j dimasukkan ke dalam *Knapsack*, dan nilai 0 berarti tidak dimasukkan

N : jumlah *item*

Lalu, dari masalah di atas, nyatakan vektor $p = [p_1, p_2, \dots, p_N]$ sebagai vektor nilai, dan vektor $w = [w_1, w_2, \dots, w_N]$ sebagai vektor berat.

Inisialisasi Parameter

Setelah diberikan permasalahan dari {0,1}-KP, sejumlah vektor solusi akan dibentuk (*generate*) secara acak sebagai himpunan pemain awal untuk algoritma SLC. Sesuai algoritma ini, nantinya vektor-vektor solusi (para pemain) tersebut akan dibagi ke dalam beberapa tim/kelompok yang akan ditandingkan/dibandingkan dalam beberapa musim pertandingan. Jumlah vektor solusi yang akan dibangun akan berjumlah

$$nP = nT \times (nF + nS) \qquad (4)$$

dengan

nP : jumlah vektor solusi atau pemain yang akan dibangun,

nT : parameter yang akan menjadi jumlah tim,

nF : parameter yang akan menjadi jumlah pemain utama,

nS : parameter yang akan menjadi jumlah pemain cadangan,

Parameter nT pada persamaan (4) adalah sebuah konstanta bilangan bulat positif yang menyatakan banyaknya kelompok yang akan membagi vektor-vektor solusi tersebut. Semakin besar nilai nT maka akan semakin banyak kemungkinan solusi yang akan dibangun. Parameter nF dan nS pada persamaan (4) adalah sebuah konstanta bilangan bulat positif yang menyatakan banyaknya vektor (pemain) dalam sebuah tim. Dari simulasi yang dilakukan Moosavian, disarankan menggunakan nilai parameter $4 \leq nT \leq 10$ dan $nF = nS = 10$ untuk simulasi komputer.

Membangun Himpunan Solusi Awal

Setelah nilai parameter ditentukan, vektor-vektor solusi (para pemain) dibangun secara acak untuk menjadi himpunan solusi awal, sebanyak jumlah pemain yang dibutuhkan dalam sebuah liga, yaitu nP. Setiap vektor solusi mempunyai elemen bernilai 0 atau 1 sejumlah N item. Vektor-vektor solusi ini berbentuk

$$P_k = [x_{k1}\ x_{k2}\ \dots x_{kN}] \qquad (5)$$

dengan

P_k : vektor solusi ke-k yang akan menjadi pemain (*player*) ke-k

x_{kj} : elemen yang bernilai 0 atau 1 untuk $k = 1, 2, \dots, nP$ dan $j = 1, 2, \dots, N$.

Selanjutnya, kekuatan pemain dihitung berdasarkan fungsi objektif dari {0,1}-KP dan sebuah fungsi penalti untuk mengeluarkan solusi tidak layak. Besar kekuatan ini menggunakan sebuah fungsi tujuan dengan formula

$$PP_k = \sum_{j=1}^{N} p_j x_{kj} - \lambda \times \max\left(\left(\frac{\sum_1^N w_j x_{kj}}{c} - 1\right), 0\right) \qquad (6)$$

dengan

PP_k : adalah kekuatan pemain (*player's power*) ke-k,

$\sum_{j=1}^{N} p_j x_{kj}$: adalah nilai fungsi objektif dari vektor solusi dari pemain ake-k yang bersangkutan, dan

λ : konstanta penalti yang ditentukan besarnya yaitu a$\lambda > \sum_{j=1}^{N} p_j$.

Pada Persamaan (6), $\lambda \times \max\left(\left(\frac{\sum_1^N w_j x_{kj}}{c} - 1\right), 0\right)$ disebut sebagai fungsi penalti. nilai $-\lambda \times \max\left(\left(\frac{\sum_1^N w_j x_{kj}}{c} - 1\right), 0\right)$ akan bernilai negatif untuk vektor yang tidak layak, dan bernilai 0 untuk vektor yang layak.

Membentuk Tim

Setelah himpunan solusi awal terbentuk, akan dibagi ke dalam beberapa tim berjumlah nT. Setiap kelompok nantinya akan beranggotakan $(nF + nS)$ vektor. Sebelum dikelompokkan, vektor-vektor (para pemain) diurutkan terlebih dahulu menurut nilai fungsi tujuannya (kekuatan pemain). Sejumlah pemain dengan nilai fungsi tertinggi akan menjadi pemain utama yang dibagi berurut dari tim ke-1 hingga tim ke-nT, dan pemain sisanya akan menjadi pemain cadangan yang dibagi berurut dari tim ke-1 hingga tim ke-nT. Prosedur pembagian tim ini dilakukan sesuai prosedur seperti pada Tabel 1 dan Tabel 2.

Tabel 1. Penempatan pemain utama di tim

F_1	P_1	sampai	P_{nF}	menjadi pemain utama	Tim_1
F_2	P_{nF+1}	sampai	P_{2nF}	menjadi pemain utama	Tim_2
⋮	⋮	⋮	⋮	⋮	⋮
F_{nF}	$P_{(nT-1)nF+1}$	sampai	P_{nTnF}	menjadi pemain utama	Tim_{nT}

Tabel 2. Penempatan pemain cadangan di tim

S_1	P_{nTnF+1}	sampai	$P_{nTnF+nS}$	menjadi pemain cadangan	Tim_1
S_2	$P_{nTnF+nS+1}$	sampai	$P_{nTnF+2nS}$	menjadi pemain cadangan	Tim_2
⋮	⋮	⋮	⋮	⋮	⋮
S_{nS}	$P_{nTnF+(nT-1)nS}$	sampai	$P_{nT(nF+nS)}$	menjadi pemain cadangan	Tim_{nT}

Prosedur di atas membagi secara urut mulai dari pemain utama dari tim 1, lalu pemain utama tim 2, hingga pemain utama tim nT. Selanjutnya, pemain yang tersisa akan dijadikan pemain cadangan, mulai dari tim 1, lalu tim 2, hingga ke tim nT. Secara lengkap, prosedur di atas akan menghasilkan komposisi seperti Tabel 3.

Komposisi tim akan berisi pemain-pemain terbaik pada Tim_1, lalu pemain-pemain terbaik selanjutnya ada pada Tim_2 dan seterusnya. Pemain utama pertama/teratas pada setiap tim (F_1) merupakan pemain dengan kekuatan/nilai fungsi yang paling besar dibanding pemain lain di tim nya. Kemudian akan dihitung SP_i dan SSP, dimana SP_i merupakan pemain bintang (*star player*) dari tim i, yang merupakan pemain teratas, yaitu F_1. Sedangkan SSP adalah pemain super (*super star player*), dimana SSP diambil dari SP_i yang mempunyai kekuatan paling besar, sehingga

$SP_i = F_1$ dari setiap tim, untuk $i = 1, 2, \dots, nT$

$SSP = SP^*$, dimana SP^* adalah SP dengan kekuatan maksimum pada populasi awal, $SSP = SP_1$

Tabel 3. Komposisi seluruh tim

	Tim_1	Tim_2	...	Tim_{nT}
F_1	P_1	P_{nF+1}	...	$P_{(nT-1)nF+1}$
F_2	P_2	P_{nF+2}	...	$P_{(nT-1)nF+2}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
F_{nF}	P_{nF}	P_{2nF}	...	$P_{nT(nF+nS)}$
S_1	P_{nTnF+1}	$P_{nTnF+nS+1}$	...	$P_{nTnF+(nT-1)nS}$
S_2	P_{nTnF+2}	$P_{nTnF+nS+2}$	...	$P_{nTnF+(nT-1)nS+1}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
S_{nS}	$P_{nTnF+nS}$	$P_{nTnF+2nS}$	...	$P_{nT(nF+nS)}$

Setelah semua telah terbagi ke beberapa kelompok, akan dihitung kekuatan setiap tim (*team's power*). Nilai ini didapat dari penjumlahan seluruh kekuatan para pemain utama dari tim tersebut. Untuk nilai $i = 1, 2, \dots, nT$, kekuatan tim ke-i dinotasikan dengan TP_i, dengan formula

$$TP_i = \sum_{k=1}^{nF} PP_k. \quad (7)$$

Memulai Liga

Setelah TP_i, SP_i, dan SSP ditentukan, langkah selanjutnya adalah melakukan liga yang berisikan pertandingan antar 2 tim. Pertandingan dimaksudkan untuk mencari vektor-vektor mana yang akan diperbarui supaya nilainya lebih optimal. Total pertandingan yang akan dilangsungkan bergantung pada jumlah tim yaitu sebanyak

$$\text{Total pertandingan} = nT(nT-1) \quad (8)$$

Formula ini di dapat dari jumlah tim seluruhnya, yaitu nT, melawan jumlah kemungkinan lawannya, yaitu $nT-1$.

Total pertandingan pada persamaan (8) akan menjadi jumlah iterasi yang akan dilakukan algoritma SLC. Untuk menambah jumlah iterasi, parameter $nSeason$ dapat ditambahkan ke dalam algoritma. Parameter $nSeason$ menyatakan jumlah musim yang akan dijalankan. Satu musimnya, pertandingan akan berjalan sebanyak pada persamaan (8). Jika musim lebih dari 1, maka total seluruh pertandingan dalam $nSeason$ sebanyak

$$\text{Total pertandingan} = nSeason \times nT(nT-1) \quad (9)$$

Pertandingan

Pada langkah ini pertandingan dilakukan berdasarkan semua pasangan tim yang mungkin di liga tersebut. Pemenang dari setiap pertandingan ditentukan secara acak melalui probabilitas dengan rumus berikut.

$$Pv(Tim_a) = \frac{TP_a}{TP_a+TP_b} \quad (10)$$

$$Pv(Tim_b) = \frac{TP_b}{TP_a+TP_b} \quad (11)$$

dengan

$Pv(Tim_a)$: probabilitas kemenangan tim a
$Pv(Tim_b)$: probabilitas kemenangan tim b
TP_a : kekuatan tim a
TP_b : kekuatan tim b

Perhatikan bahwa jumlah $Pv(Tim_a) + Pv(Tim_b) = \frac{TP_a}{TP_a+TP_b} + \frac{TP_b}{TP_a+TP_b} = 1$.

Pemenang ditentukan secara acak. Tim dengan kekuatan yang lebih besar mempunyai peluang yang lebih tinggi untuk menang. Kemudian setelah tim pemenang ditentukan, para pemain utama dan cadangan dari tim pemenang tersebut akan mengalami pembaruan dengan Operator Imitasi.

Operator Imitasi

Pemain utama dan cadangan dari tim pemenang akan berusaha menjadi lebih baik lagi dengan cara mengimitasi/mencontoh SSP dan SP dari timnya sendiri. Vektor solusi yang bersangkutan akan 'bergerak' ke arah vektor solusi terbaik dari liga (SSP) atau vektor solusi terbaik dari timnya (SP_i) dengan prosedur berikut.

1) Pada langkah pertama, sebuah vektor baru akan dibuat untuk merepresentasikan pemain mengimitasi SSP dengan formula sebagai berikut

$$P_{new}(j) = P_k(j) + \tau\big(SSP(j) - RP1(j)\big) \qquad (12)$$

di mana

$P_{new}(j)$: variabel keputusan ke-j untuk vektor yang baru
$P_k(j)$: variabel keputusan ke-j dari vektor (pemain) ke-k
$\tau \sim U(0.2, 0.8)$: bilangan acak berdistribusi Uniform (0.2, 0.8)
$SSP(j)$: variabel keputusan ke-j dari SSP
$RP1(j)$: variabel keputusan ke-j dari suatu pemain acak dari tim tersebut

Jika formula di atas menghasilkan vektor yang memiliki nilai fungsi lebih baik, maka vektor yang baru akan menggantikan vektor yang lama. Jika tidak, maka dibuat vektor baru lagi dengan ketentuan sebagai berikut.

2) Pada langkah kedua, sebuah vektor baru akan dibuat untuk merepresentasikan pemain mengimitasi SP timnya dengan formula sebagai berikut

$$P_{new}(j) = P_k(j) + \tau\big(SP(j) - RP1(j)\big) \qquad (13)$$

di mana

$P_{new}(j)$: variabel keputusan ke-j untuk vektor yang baru.
$P_k(j)$: variabel keputusan ke-j dari vektor (pemain) ke-k .
$\tau \sim U(0.2, 0.8)$: bilangan acak berdistribusi Uniform (0.2, 0.8)
$SP(j)$: variabel keputusan ke-j dari SP tim pemenang
$RP1(j)$: variabel keputusan ke-j dari suatu pemain acak dari tim tersebut

Jika formula di atas menghasilkan vektor yang memiliki nilai fungsi lebih baik, maka vektor yang baru akan menggantikan vektor yang lama. Jika tidak, maka dibuat vektor baru lagi dengan ketentuan sebagai berikut.

3) Pada langkah ketiga, sebuah vektor baru akan dibuat untuk merepresentasikan pemain yang mengimitasi pemain lainnya secara acak, dengan formula sebagai berikut.

$$P_{new}(j) = P_k(j) + \tau\big(RP1(j) - RP2(j)\big) \qquad (14)$$

di mana

$P_{new}(j)$: variabel keputusan ke-j untuk vektor yang baru.
$P_k(j)$: variabel keputusan ke-j dari vektor (pemain) ke-k .
$\tau \sim U(0.2, 0.8)$: bilangan acak berdistribusi Uniform (0.2, 0.8)
$RP1(j)$: variabel keputusan ke-j dari suatu pemain acak dari tim tersebut
$RP2(j)$: variabel keputusan ke-j dari suatu pemain acak yang lain dari tim tersebut

Jika formula di atas menghasilkan vektor yang memiliki nilai fungsi lebih baik, maka vektor yang baru akan menggantikan vektor yang lama. Jika tidak, maka ada perubahan pada vektor lama.

Perbarui Tim

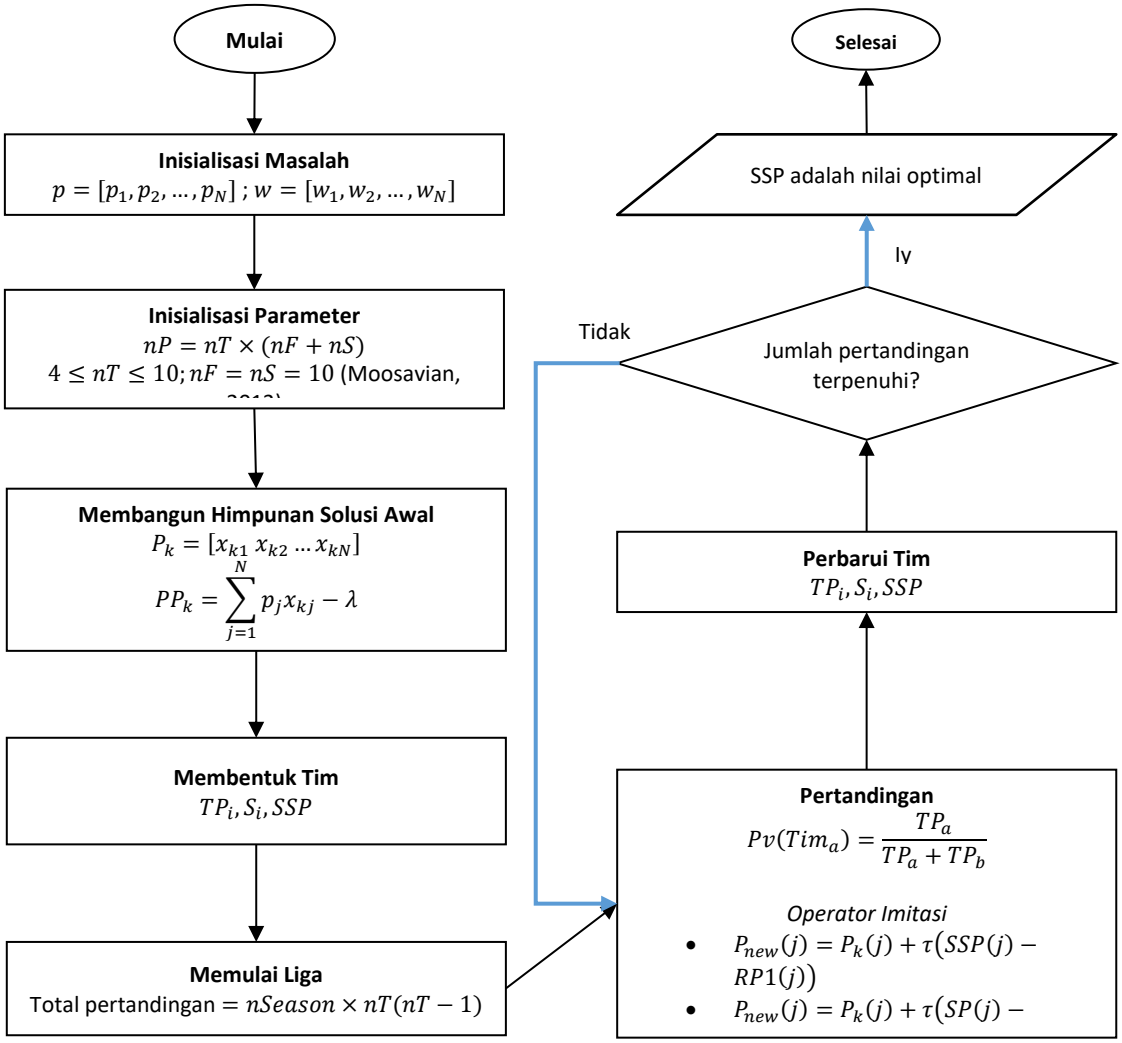
Setelah Operator Imitasi bekerja, selanjutnya adalah memperbarui TP_i, SP_i, dan SSP. TP_i mengalami pembaruan dengan formula yang sama, yaitu $TP_i = \sum_{k=1}^{nF} PP_k$. Sedangkan SP_i dan SSP akan diperiksa dengan nilai SP dan SSP sebelumnya. Jika nilai fungsi dari vektor yang baru lebih baik, makan akan dilakukan pembaruan. Jika tidak lebih baik, maka vektor SP dan SSP tetap.

Memeriksa Kriteria Berhenti

Langkah selanjutnya adalah memeriksa apakah semua tim telah bertanding dan jumlah pertandingan telah terpenuhi atau belum. Jika sudah, maka lanjut ke langkah selanjutnya. Jika belum, ulangi langkah 6 hingga langkah 8 kembali sampai semua pertandingan dilaksanakan.

Penentuan Hasil

Setelah seluruh pertandingan dilaksanakan, maka SSP terakhir akan menjadi vektor solusi optimal untuk masalah {0,1}-KP yang telah diinisialisasi di awal. Berdasarkan prosedur algoritma SLC di atas, alur kerja algoritma ini dapat dinyatakan pada Gambar 1.



Gambar 1. Alur kerja Algoritma SLC

HASIL DAN PEMBAHASAN

Akan dilakukan simulasi untuk sepuluh masalah {0,1}-KP. Data berisi sejumlah item dengan keterangan nilai dan berat, serta kapasitas Knapsack dari masalah tersebut yang dapat dilihat pada Tabel 4.

Tabel 4. Data 10 masalah {0,1}-KP

Masalah	Jumlah item	Kapasitas Knapsack (c)	Daftar nilai (p) dan berat (w)
f_1	4	20	p =[9 11 13 15] w =[6 5 9 7]
f_2	4	11	p =[6 10 12 13] w =[2 4 6 7]
f_3	5	80	p =[33 24 36 37 12] w =[15 20 17 8 31]
f_4	7	50	p =[70 20 39 37 7 5 10] w =[31 10 20 19 4 3 6]
f_5	10	269	p =[55 10 47 5 4 50 8 61 85 87] w =[95 4 60 32 23 72 80 62 65 46]
f_6	10	60	p =[20 18 17 15 15 10 5 3 1 1] w =[30 25 20 18 17 11 5 2 1 1]
f_7	15	375	p =[0.125126 19.330424 58.500931 35.029145 82.284005 17.410810 7150142 30.399487 9.140294 14.731285 98.852504 11.908322 0.891140 53.166295 60.176397] w =[56.358531 80.874050 47.987304 89.596240 74.660482 85.894345 51.353496 1.498459 36.445204 16.589862 44.569231 0.466933 37.788018 57.118442 60.716575]
f_8	20	878	p =[44 46 90 72 91 40 75 35 8 54 78 40 77 15 61 17 75 29 75 6] w =[92 4 43 83 84 68 92 82 6 44 32 18 56 83 25 96 70 48 14 58]
f_9	20	879	p =[91 72 90 46 55 8 35 75 61 15 77 40 63 75 29 75 17 78 40 44] w =[84 83 43 4 44 6 82 92 25 83 56 18 58 14 48 70 96 32 68 92]
f_{10}	23	10000	p =[981 980 979 978 977 976 487 974 970 485 485 970 970 484 484 976 974 482 963 961 959 958 957] w =[983 982 981 980 979 978 488 976 972 486 486 972 972 485 485 969 966 483 964 963 961 958 959]

[Sumber : Zou *et al*. (2011)]

Sebelum penyelesaian {0,1}-KP menggunakan algoritma SLC, akan ditampilkan terlebih dahulu solusi optimal dari sepuluh masalah {0,1}-KP yang diselesaikan oleh Zou

et al (2011) menggunakan algoritma Novel Global Harmony Search (NGHS), yang akan disajikan pada Tabel 5.

Tabel 5. Penyelesaian 10 masalah {0,1}-KP dengan algoritma NGHS

Masalah	Jumlah item	Solusi optimal	Nilai fungsi
f_1	4	[1 1 0 1]	35
f_2	4	[0 1 0 1]	23
f_3	5	[1 1 1 1 0]	130
f_4	7	[1 0 0 1 0 0 0]	107
f_5	10	[0 1 1 1 0 0 0 1 1 1]	295
f_6	10	[0 0 1 0 1 1 1 1 0 0]	50
f_7	15	[0 0 1 0 1 0 1 1 0 1 1 1 0 1 1]	481.0694
f_8	20	[1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 1 1]	1024
f_9	20	[1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 0 1 1 1]	1025
f_{10}	23	[1 1 1 1 1 1 1 1 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0]	9767

Dalam penyelesaian {0,1}-KP menggunakan algoritma SLC, akan digunakan nilai parameter seperti pada Tabel 6.

Tabel 6. Parameter yang digunakan dalam simulasi

	Nilai Parameter
$nSeason$	1
nT	4
nF	10
nS	10

Dari kesepuluh masalah pada Tabel 4, dengan menggunakan beberapa parameter yang telah disebutkan pada Tabel 6, kemudian akan dilakukan proses pencarian solusi sesuai dengan alur algoritma SLC yang telah diimplementasikan dalam perangkat lunak tersebut dan hasilnya disajikan pada Tabel 7.

Tabel 7. Hasil simulasi algoritma SLC

Masalah	Jumlah item	Solusi yang didapat	Nilai fungsi	Waktu komputasi (dalam detik)
f_1	4	[1 1 0 1]	35	0.0859999656677
f_2	4	[0 1 0 1]	23	0.0929999351501
f_3	5	[1 1 1 1 0]	130	0.0929999351501
f_4	7	[1 0 0 1 0 0 0]	107	0.0989999771118
f_5	10	[0 1 1 1 0 0 0 1 1 1]	295	0.128999948502
f_6	10	[0 0 1 1 0 1 1 1 1 1]	52	0.1890001297
f_7	15	[0 0 1 1 1 0 1 1 0 0 1 1 0 1 0]	441.190831	0.125
f_8	20	[0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1]	977.0	0.151000022888
f_9	20	[1 1 1 1 1 1 0 1 1 0 1 1 1 1 0 1 1 1 1 0]	963.0	0.132999897003

f_{10}	23	[1 1 1 1 1 1 1 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0]	9750	0.194000005722

Dari Tabel 5 dan Tabel 7, solusi optimal dan nilai fungsi dari algoritma NGHS dan SLC akan dibandingkan. Dapat dilihat bahwa solusi NGHS dan SLC memberikan nilai yang sama pada f_1 sampai f_5.

Pada f_6 SLC memberikan nilai fungsi yang lebih besar dari NGHS. NGHS memberikan nilai fungsi 50 dan SLC memberikan nilai fungsi 52. Kemudian pada f_7 sampai f_{10} NGHS menampilkan hasil yang lebih baik dari hasil yang ditampilkan oleh SLC.

Selanjutnya, akan disimulasikan kembali f_1 sampai f_{10} menggunakan algoritma SLC dengan nilai parameter yang berbeda, seperti pada tabel 8.

Tabel 8. Parameter yang digunakan pada simulasi kedua

Nilai Parameter	
$nSeason$	4
nT	7
nF	10
nS	10

Kemudian, hasil simulasi SLC menggunakan nilai parameter di atas diberikan pada Tabel 9.

Tabel 9. Hasil simulasi kedua algoritma SLC

Masalah	Jumlah item	Solusi yang didapat	Nilai fungsi	Waktu komputasi (dalam detik)
f_1	4	[1 1 0 1]	35	0.625
f_2	4	[0 1 0 1]	23	0.631000041962
f_3	5	[1 1 1 1 0]	130	0.744999885559
f_4	7	[1 0 0 1 0 0 0]	107	0.933000087738
f_5	10	[0 1 1 1 0 0 0 1 1 1]	295	1.35899996758
f_6	10	[0 0 1 1 0 1 1 1 1 1]	52	1.14400005341
f_7	15	[0 0 1 0 1 0 1 1 0 1 1 1 0 1 1]	481.069368	1.5490000248
f_8	20	[1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 1 1]	1024	1.86000013351
f_9	20	[1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 0 1 1 1]	1025	1.87699985504
f_{10}	23	[1 1 1 1 1 1 1 1 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0]	9767	2.20700001717

Dari hasil simulasi pada Tabel 9, algoritma SLC memberikan solusi yang sama dengan solusi yang didapatkan dengan algoritma NGHS, kecuali pada f_6. Jika dilakukan perhitungan secara manual, solusi optimal yang didapatkan dari NGHS memiliki total berat

$$0 \times 30 + 0 \times 25 + 1 \times 20 + 0 \times 18 + 1 \times 17 + 1 \times 11 + 1 \times 5 + 1 \times 2 + 0 \times 1 + 0 \times 1 = 55.$$

Sedangkan solusi optimal yang didapatkan SLC pada f_6 memiliki total berat

$$0 \times 30 + 0 \times 25 + 1 \times 20 + 1 \times 18 + 0 \times 17 + 1 \times 11 + 1 \times 5 + 1 \times 2 + 1 \times 1 + 1 \times 1 = 58.$$

Kedua algoritma menghasilkan solusi optimal yang memiliki total berat yang tidak melebihi kapasitas knapsack pada f_6.

Dari hasil simulasi pertama dan kedua algoritma SLC, didapatkan bahwa nilai parameter mempengaruhi ketercapaian solusi optimal pada sebuah masalah. Makin besar nilai parameter, makin besar kemungkinan solusi optimal tercapai, tetapi waktu komputasi yang dibutuhkan juga makin besar.

SIMPULAN

Berdasarkan simulasi, didapatkan bahwa algortima *Soccer League Competition* (SLC) dapat diterapkan untuk menyelesaikan {0,1}-*Knapsack Problem* ({0,1}-KP). Berdasarkan hasil simulasi pertama pada Tabel 7 dan kedua pada Tabel 9, terlihat bahwa perubahan nilai parameter mempengaruhi tercapainya solusi optimal yang dapat diperoleh. Selain itu, waktu komputasi juga bergantung pada nilai parameter tersebut. Makin besar nilai parameter, makin banyak waktu komputasi yang dibutuhkan.

Berdasarkan hasil simulasi pada masalah f_6 pada kedua simulasi, yaitu pada Tabel 7 dan Tabel 9, terlihat bahwa algoritma SLC memberikan nilai fungsi yang berbeda dari algoritma NGHS yang ditampilkan pada Tabel 5. Hal ini mungkin terjadi karena perbedaan data yang dikarenakan oleh *human error*.

DAFTAR PUSTAKA

Kellerer, H., Pferschy, U., & Pisinger, D. 2004. *Knapsack Problems.* New York: Springer-Verlag Berlin Heidelberg.

Li, Z., & Li, N. 2009. A novel multi-mutation binary particle swarm optimization for 0/1 knapsack problem. *Chinese Control and Decision Conference* (pp. 3090-3095). New Jersey: IEEE Press.

Lin, F.-T. (2008, February 16). Solving the knapsack problem with imprecise weight coefficients using genetic algorithms. *European Journal of Operational Research, 185*(1), 133-145.

Liu, Y., & Liu, C. 2009. A Schema-Guiding Evolutionary Algorithm for 0-1 Knapsack Problem. *International Association of Computer Science and Information Technology - Spring Conference* (pp. 160-164). IEEE.

Martello, S., & Toth, P. 1990. *Knapsack Problems : Algorithms and Computer Implementations.* West Sussex: John Wiley & Sons Ltd.

Moosavian, N. 2015. Soccer league competition algorithm for solving knapsack problems. *Swarm and Evolutionary Computation, 20*, 14-22.

Pisinger, D. 1995. Algorithms for Knapsack Problems. Copenhagen: Dept. of Computer Science, University of Copenhagen.

Shi, H. 2006. Solution to 0/1 Knapsack Problem Based on Improved Ant Colony Algorithm. *International Conference on Information Acquisition* (pp. 1062-1066). Shandong: IEE.

Taha, H. A. 2007. *Operations Research : An Introduction* (8 ed.). New Jersey: Pearson Education, Inc.

Winston, W. L. 2003. *Operations Research: Applications and Algorithms.* Kentucky: Cengage Learning.

Zou, D., Gao, L., Li, S., & Wu, J. 2011. Solution to 0/1 Knapsack Problem by a Novel Global Harmony Search Algorithm. *Applied Soft Computing, 11*(2), 1556-1564.