دانشکده مهندسی

گروه مهندسی صنایع

پروژه درس تصمیم گیری چند معیاره

عنوان: کد نویسی یک مسئله زمانبندی flow shop چندهدفه با استفاده از الگوریتم انجماد تدریجی

استاد راهنما: جناب دکتر امیرسامان خیرخواه

نگارش: مینا فرجی امیری

۹۴۱۳۴۸۷۰۰۲

زمستان ۹۴

کلیات موضوع

زمانبندی پروسه ی تولید یک برنامه است و برنامه به صورت کلی اتفاق افتادن مسائل و افقی برای زمان انجام فعالیت های خاص را به ما می دهد. به صورت کلی مسائل زمانبندی به دو مرحله تقسیم می شوند؛ در اولین مرحله توالی برنامه ریزی می شود و چگونگی انتخاب وظیفه بعدی مشخص می شود. در دومین مرحله برنامه ریزی زمان شروع و پایان هر یک از وظایف صورت می گیرد.

هروظیفه بر اساس اطلاعاتی مانند منابع مورد نیاز، طول اجرا، زودترین زمانی که می تواند شروع شود و زمان تحویل آن توصیف می شود. همچنین هر محدودیت تکنولوژیکی مانند محدودیت پیش نیازی نیز باید مشخص شود. اطلاعات مربوط به منابع و وظایف، یک مسئله زمانبندی را مشخص می کند و راه حل این مسئله سخت می باشد. در فضای تولید به منابع ماشین و به وظایف کارها گفته می شود.

فرض کنید n کار باید بر روی m ماشین پردازش شوند. محدودیت های تکنولوژیکی باعث می شود تا هر کار توسط ماشین در یک صف مشخص انجام شود که به این مسئله جریان کارگاهی flowshop گفته می شود. بنابراین در جریان تولید کارها در یک صف مشخص از ماشین ها می گذرند. مسئله زمان بندی جریان کارگاهی Flow shop در زمان وجود یک جریان پیوسته از کارهای تولیدی برروی چند ماشین مطرح می گردد.

مدت زمانی که طول می کشد تا تجهیزات در حالت دلخواه برای پردازش کار قرار گیرند را زمان راه اندازی می گویند. این زمان به کار انجام شده و کار در انتظار وابسته است. در واقع تغیرات زمان آماده سازی بر اساس هر توالی معیاری برای ارزیابی زمانبندی ایجاد می کند. با رفتن هر کار از ماشینی به ماشین دیگر این زمان تغیر می کند. زمان آماده سازی وابسته به توالی SDST یکی از موضوعات جدید و واقعی در تعمیم مسئله جریان کارگاهی است.

در دنیای واقعی مسائل زمانبندی طبیعتا چند هدفه هستند و به طور همزمان باید این اهداف ارضا شوند. توابع هدف متعددی در مسئله زمانبندی وجود دارد که از آن جمله می توان به زمان تکمیل، تاخیر کلی، بیشترین تاخیر، تعداد کارهای دارای تاخیرو... اشاره نمود.

رویکرد های متفاوتی برای مسائل چند هدفه وجود دارد چهار رویکرد اصلی به صورت زیر است روش های همزمان (رویکرد پارتو)، روش وزن دهی اهداف، روش بهینه سازی سلسله مراتبی، روش برنامه ریزی آرمانی

روش وزن دهی به اهداف یکی از ساده ترین و پرکاربردترین رویکردهای برخورد با مسائل چند هدفه است. در این روش فرض می شود شرایط استقلال مطلوبیتی اهداف به گونه ای برقرار است که می توان جمع وزنی توابع هدف را به عنوان تقریبی از تابع مطلوبیت در نظر گرفت.

از آنجایی که اثبات شده است مسئله زمانبندی جز مسائل NP-HARD است بنابراین برای حل آن می توان از الگوریتم های فرا ابتکاری استفاده کرد.

الگوریتم تبرید شبیه‌سازی شده (Simulated Annealing (SA، یک الگوریتم بهینه‌سازی فراابتکاری ساده و اثربخش در حل مسائل بهینه‌سازی است. تکنیک تبرید تدریجی، به وسیله ی متالورژیست‌ها برای رسیدن به حالتی که در آن ماده جامد، به خوبی مرتب و انرژی آن کمینه شده باشد، استفاده می‌شود. این تکنیک شامل قرار دادن ماده در دمای بالا و سپس کم کردن تدریجی این دماست. در روش شبیه سازی تبریدی، هر نقطه s در فضای جستجو مشابه یک حالت از یک سیستم فیزیکی است، و تابع E(s) که باید کمینه شود، مشابه با انرژی داخلی سیستم در آن حالت است. در این روش، هدف انتقال سیستم از حالت اولیه دلخواه، به حالتی است که سیستم در آن کمترین انرژی را داشته باشد.

برای حل یک مسئله‌ی بهینه‌سازی، الگوریتم SA ابتدا از یک جواب اولیه شروع می‌کند و سپس در یک حلقه تکرار به جواب‌های همسایه حرکت می‌کند. اگر جواب همسایه بهتر از جواب فعلی باشد، الگوریتم آن را به عنوان جواب فعلی قرار می‌دهد (به آن حرکت می‌کند)، در غیر این صورت، الگوریتم آن جواب را با احتمال exp(-ΔE/T) به عنوان جواب فعلی می‌پذیرد. در این رابطه ΔE تفاوت بین تابع هدف جواب فعلی و جواب همسایه‌است و T یک پارامتر به نام دما است. در هر دما، چندین تکرار اجرا می‌شود و سپس دما به آرامی کاهش داده می‌شود. در گام‌های اولیه دما خیلی بالا قرار داده می‌شود تا احتمال بیشتری برای پذیرش جواب‌های بدتر وجود داشته باشد. با کاهش تدریجی دما، در گام‌های پایانی احتمال کمتری برای پذیرش جواب‌های بدتر وجود خواهد داشت و بنابراین الگوریتم به سمت یک جواب خوب همگرا می‌شود.
همسایه‌های یک حالت (جواب)، حالت‌های جدیدی از مسئله هستند که با تغییر در حالت کنونی و با توجه به روشی از پیش تعیین شده ایجاد می‌شوند.

کد استفاده شده برای حل این مسئله در زیر توضیح داده می شود.

- ✓ اطلاعات مسئله ی جریان کارگاهی با زمان آماده سازی وابسته به توالی در یک ام فایل متلب از کد زیر برای تولید مساله استفاده شده است.

```
clc
clear
```

تعداد کارها برابر ده و تعداد ماشین ها پنج در نظر گرفته شده است.

```
n=10; % number of job
m=5; % number of machine
```
زمان پردازش هرکار روی هر ماشین عددی تصادفی بین یک تا صد در ماتریسی با ان سطر و ام ستون درنظر گرفته شده است.
```
p=randi([1 100],n,m); % processing time
```
زمان آماده سازی وابسته به توالی نیز به تعداد ماشین ها ماتریس هایی با ان سطر و ان ستون که نشان دهنده زمان اماده سازی کار آیم روی ماشین ام اگر قبل از آن کار جیم تمام شده باشد است.
```
for i=1:m
s{i}=randi([0 9],[n,n]); % sequence dependent set up time
end
```
زمان تحویل هر کار نیز عددی تصادفی بین ۱۱۰ تا ۵۰۰ در آرایه ای با ان سطر ذخیره می شود.
```
d=randi([110 500],n,1);  % due date of each job
save data
```

✓ الگوریتم انجماد تدریجی در یک ام فایل متلب به صورت زیر آمده است.

```
clc;
clear;
close all;

%% parametere of Simulated Annealing
```
پارامترهای الگوریتم انجماد تدریجی شامل تعداد حلقه ها، تعداد نقاط آزمایش در هر حلقه، تعداد راه حل های پذیرفته شده، احتمال پذیرش جواب بد در شروع الگوریتم، احتمال پذیرش جواب بد در پایان، دمای ابتدایی، دمای انتهایی، میزان کاهش دما بعد از هر حلقه
```
nc = 10; % Number of cycles
nt = 5; % Number of trials per cycle
na = 0.0; % Number of accepted solutions
p1 = 0.7; % Probability of accepting worse solution at the start
p10 = 0.001; % Probability of accepting worse solution at the end
t1 = -1.0/log(p1); % Initial temperature
t10 = -1.0/log(p10); % Final temperature
frac = (t10/t1)^(1.0/(nc-1.0)); % Fractional reduction every cycle

%% data of problem
```
اطلاعات خود مسئله در این قسمت از ام فایل قبلی خوانده می شود همچنین وزن هر یک از توابع هدف در نظر گرفته شده در این مسئله نیز در این قسمت وارد می شود.
```
a=load('data.mat');
nm=a.m; % number of machine
nj=a.n; % number of jobs
p=a.p; % processing time
d=a.d; % due date of each job
% setup time
for i=1:nm
s{i}=a.s{i};
end
s=cell(nm);
% weight of each objective
a1=0.2; % tardiness
a2=0.1; % makespan
a3=0.4; % earliness
a4=0.3; % number of tardy jobs
```

```
%% Initialization
```
آماده سازی

```
x = zeros(nc,1);
```
آرایه ای برای نگهداری جواب های هر حلقه

```
fs = zeros(nc,1);
```
آرایه ای برای نگهداری تابع هدف هر حلقه

```
seq = zeros(nj,1);
```
آرایه ای برای تولید توالی کارها

```
i = zeros(nj,nm);
```
ماتریسی با ابعاد تعداد کار و ماشین برای زمان بیکاری کار بر روی ماشین

```
c = zeros(nj,nm);
```
ماتریسی با ابعاد تعداد کار و ماشین برای زمان اتمام کار بر روی ماشین

تولید تصادفی اعداد بین یک تا ان جی برای جواب اولیه

```
x_start =randperm(nj);% initial solution
xc = x_start;
```
جواب کنونی

```
na = na + 1.0;% number of accepted answers
```
مقدار تابع هدف به ازای جواب اولیه

```
fc = fitness(xc,nm,nj,p,s,d,a1,a2,a3,a4);% Current best results so far
t = t1;% Current temperature
DeltaE_avg = 0.0;% DeltaE Average

%% main loop
```
حلقه اصلی الگوریتم

```
for i=1:nc
    disp(['Cycle: ',num2str(i),' with Temperature: ',num2str(t),' xc: ' ,
num2str(xc)]);
    for j=1:nt
```
تولید همسایه ی جواب فعلی

```
        xj = arr_change(xc) ; % Generate new trial points
        disp(['trial: ',num2str(j),' with trial point: ',num2str(xj)])
```
قدرمطلق تفاوت بین تابع هدف و همسایگی

```
        DeltaE = abs(fitness(xj,nm,nj,p,s,d,a1,a2,a3,a4)-fc);
```
اگر جواب همسایگی بدتر از جواب فعلی باشد

```
        if (fitness(xj,nm,nj,p,s,d,a1,a2,a3,a4)>fc)
            % Initialize DeltaE_avg if a worse solution was found
```
در اولین تکرار از اولین حلقه

```
            if (i==1 && j==1)%   on the first iteration
                DeltaE_avg = DeltaE;
            end
            % objective function is worse
```
احتمال پذیرش

```
            pp = exp(-DeltaE/(DeltaE_avg * t));% generate probability of
acceptance
           % determine whether to accept worse point
```
تولید عدد تصادفی و مقایسه با احتمال پذیرش برای قبول یا رد جواب

```
            if (rand()<pp)
                accept = true;% accept the worse solution
            else
                accept = false;% don't accept the worse solution
            end
        else
            accept = true;% objective function is lower, automatically
accept
        end
```
به روز رسانی جواب

```
        if (accept==true)
```

```
            xc = xj;% update currently accepted solution
            fc =fitness(xj,nm,nj,p,s,d,a1,a2,a3,a4);
            na = na + 1.0;% increase number of accepted solutions
            DeltaE_avg = (DeltaE_avg * (na-1.0) +  DeltaE) / na;% update
DeltaE_avg
        end
    end
    % Record the best x values at the end of every cycle
```

نگهداری جواب هر حلقه

```
    xi = xc;
    fs(i) = fc;
```

کاهش دما در آخر حلقه

```
    t = frac * t;% Lower the temperature for next cycle
end

%% results
```

نمایش جواب ها

```
disp(['Best solution: ',num2str(xi)])
disp(['Best objective: ',num2str(fs(i))])

fig=figure(1);
plot(xi,'b.-');
xlabel('cycles');
ylabel('best solution');
grid on;
fig=figure(2);
plot(fs,'r.-');
xlabel('cycles');
ylabel('best objective');
grid on;
```

- ✓ تولید همسایه ی یک نقطه در ام فایل زیر آمده است.

```
function xc=arr_change(xc)
[y,x]=size(xc);
```

تعداد اعضای توالی اصلی

```
p=randperm(x);
```

تولید اعداد تصادفی بین یک و عدد ایکس

انتخاب دو عدد اول در آرایه ی پی و جا به جایی جایگاه این دو عدد در توالی اصلی

```
temp=xc(p(1));
xc(p(1))=xc(p(2));
xc(p(2))=temp;
 end
```

- ✓ تابع هدف چهارهدفه ی مسئله جریان کارگاهی با روش میانگین وزنی در یک ام فایل به صورت زیر آمده است.

```
function scores = fitness(x,nm,nj,p,s,d,a1,a2,a3,a4)
scores = zeros(size(x,1),1);
seq=x;
```

توالی داده شده کارها

```
pr=0; %precedence restrictions
```

محدودیت پیش نیازی

```
ta=0; %total tardiness
```

تاخیر کلی

```
el=0; %total earliness
```

تعجیل کلی

```
ntj=0; %number of tardy jobs تعداد کارهای با تاخیر
for k=1:length(seq)
j=seq(k);
for m= 1:nm
if m== 1ماشین اول
i(j,1)=0;زمان بیکاری کار جی روی ماشین یک
    if pr==0 اگر محدودیت پیش نیازی نداشته باشد
    c(j,1)=p(j,1);زمان تکمیل کار جی روی ماشین یک برابر است با زمان پردازشش
    else
    c(j,1)=c(pr,1)+s{1}(pr,j)+p(j,1); زمان تکمیل کار جی روی ماشین یک برابر زمان تکمیل
پیشنیازی آن به اضافه زمان آماده سازی وابسته به توالی روی ماشین یک به اضافه زمان پردازش کار جی روی ماشین یک
    end
else سایر ماشین ها به جز ماشین اول
محاسبه زمان بیکاری یا تلف شده(قرار گرفتن کارها در صف)
    if pr==0محدودیت پیشنیازی وجود ندارد
        if i(j,m-1)<0 اگر صفی بر روی ماشین قبلی موجود نباشد(زمان از دست رفته نداشته باشیم)
        i(j,m)=i(j,m-1)-p(j,m-1); زمان بیکاری کار جی روی ماشین ام برابر است بامجموع زمان بیکاری کار جی روی
ماشین قبلی به اضافه زمان پردازش آن
        else
        i(j,m)=-p(j,m-1);
        end
    else با وجود محدودیت پیشنیازی
        if i(j,m-1)<0
        i(j,m)=c(pr,m)-c(pr,m-1)+s{m}(pr,j)-s{m-1}(pr,j)-p(j,m-1)+i(j,m-1);
زمان بیکاری کار جی روی ماشین ام برابر است با زمان تکمیل پیشنیازها روی ماشین ام منهی زمان تکمیل پیشنیازی روی
ماشین قبلی به علاوه اختلاف زمان آماده سازی پیش نیازی روی ماشین ام و قبلی منهی زمان پردازش ماشین قبلی به علاوه
زمان بیکاری ماشین قبلی
        else
        i(j,m)=c(pr,m)-c(pr,m-1)+s{m}(pr,j)-s{m-1}(pr,j)-p(j,m-1);
        end
    end
    if i(j,m)>=0 اگر زمان بیکاری مقدار مثبت داشته باشد یعنی بیکاری داشته باشیم
    c(j,m)=c(j,m-1)+p(j,m)+i(j,m);
    else اگر زمان بیکاری یا تلف شده نداشته باشیم
    c(j,m)=c(j,m-1)+p(j,m);
    end
end
end
if c(j,nm)>d(j) اگر زمان تکمیل کار جی روی آخرین ماشین بزرگتر از موعد تحویل باشد تعداد کارهای عقب افتادهو
تاخیر افزایش می یابد
    ntj=ntj+1;
    ta=ta+(c(j,nm)-d(j));
else
    ta=ta+0;
end
if d(j)>c(j,nm) اگر موعد تحویل بزرگتر از زمان تکمیل باشد تعجیل افزایش می یابد
    el=el+(d(j)-c(j,nm));
else
    el=el+0;
end
```

پیشنیازی را برابر جی که کار قرارگرفته در نوبت کا ام است قرار می دهد ;pr=j

```
end
scores = a1*ta+a2*c(j,nm)+a3*el+a4*ntj
scores
```