サクラスケジューラの理論的背景（ラグランジュ緩和法）

桜ロジック

2011 年 9 月 30 日

概 要

サクラスケジューラには、ラグランジュ緩和法を使った 4 種類のスケジューリング・ロジック、LR_G_Scheduler、LR_S_Scheduler、LR_H_Scheduler、および LR_T_Scheduler が既定で組み込まれています。この文書ではそれらのスケジューリング・ロジックの理論的な背景を詳述します。サクラスケジューラが扱うスケジューリング問題は、繰り返し加工を許すフレキシブルジョブショップ問題（FJSP：Flexible Job Shop Problem）です。ここで繰り返し加工とは、ジョブがその技術的順序で処理されていく過程で同じ作業（工程）を繰り返すことを意味します。ここで、ジョブの原材料が先頭工程に到着する時刻は決まっており、また各作業（工程）を処理する機械が故障や点検のために止まっている場合も考慮します。目的関数は投入進みの二乗と納期遅れの二乗の和の平均、または納期進みの二乗と納期遅れの二乗の和の平均です。ラグランジュ緩和法の実行可能化の手法として貪欲法、探索法、および Giffler & Thompson 法の 3 種類を適用しています。貪欲法はその最適度（反復改善過程の初期解と近似最適解の目的関数の改善度）は探索法と比較してよいですが、計算時間がジョブの密度（スケジュールの込み具合）に強く依存するという欠点を持ちます。一方、探索法は貪欲法と比較してその最適度は多少劣りますが、計算時間が少なくて済むという利点があります。また、Giffler & Thompson 法は他の実行可能化の手法と比較して、ジョブ数が多くなると最適度も劣り、より多くの計算時間を必要とするという欠点があります。スケジューリング・ロジック LR_G_Scheduler は、実行可能化に貪欲法のみを使い、LR_S_Scheduler は実行可能化に探索法のみを使います。また、LR_H_Scheduler は双対問題を解く間は探索法を使い、最後に貪欲法を使って近似最適解を求めます。そして、LR_T_Scheduler は実行可能化に Giffler & Thompson 法を使います。ラグランジュ緩和法は、遺伝的アルゴリズムと比較して計算時間も大幅に少なく、ジョブの密度が高い場合に改善率もよいです。

1　問題の定式化

サクラスケジューラに組み込まれているラグランジュ緩和法を使ったスケジューリング・ロジックでは、投入進みと納期遅れの二乗の和の平均、または納期進みと納期遅れの二乗の和の平均（進み-遅れペナルティ）を最小にするよう、N 個のジョブを M 台の機械で処理するスケジュールを作成する問題を扱う。ジョブはそれぞれに異なった納期を持ち、特定の技術的順序で処理されなければならない工程から成る。各工程は複数の機械で処理が可能であり、スケジューリングの過程で目的関数値が最小となるように工程を処理する機械が決定される。また、ジョブが同じ機械で複数回処理されることを許す。以下の変数がスケジューリング問題の定式化で使われる。

N	ジョブ数、
n_i	ジョブ i の工程の数、
w_i	ジョブ i の重み、
M	機械の数、
$\mathcal{M}_{i,j}$	ジョブ i の工程 j を処理可能な機械（複数台）、
D_i	ジョブ i の納期、
R_i	ジョブ i の先頭工程の集合、
G_i	ジョブ i の終了工程の集合、
$P_{i,j}$	ジョブ i の工程 j の先行工程の集合、
$t_{i,j,m}$	ジョブ i の工程 j の機械 m での加工時間、
$c_{i,j}$	ジョブ i の工程 j の完了時刻、
$b_{i,j}$	ジョブ i の工程 j の開始時刻、
$a_{i,j}$	ジョブ i の先頭工程 $j \in R_i$ に原材料が到着する時刻、
$s_{k,l}$	工程 k と後続工程 l の間の余裕時間、
H	考慮する時刻数。

技術的順序は無閉路有向グラフを成し、複数の工程で始まり、複数の工程で終わり得る。工程 k と工程 l の間には、搬送や待機時間を表す余裕時間 $s_{k,l}$ を設定する。ジョブ i の開始工程の集合を R_i、そして終了工程の集合を G_i とすると、ジョブの投入進み $I_{i,r}$、納期進み $E_{i,g}$ と納期遅れ $T_{i,g}$ はそれぞれ次のように定義される。

$$I_{i,r} = \max\{0, r_{i,r} - b_{i,r}\}、r \in R_i \tag{1}$$

$$E_{i,g} = \max\{0, d_{i,g} - c_{i,g}\}、g \in G_i \tag{2}$$

$$T_{i,g} = \max\{0, c_{i,g} - d_{i,g}\}、g \in G_i \tag{3}$$

ここで、$r_{i,r}$ と $d_{i,g}$ は、ジョブの納期 D_i を所与とする無限能力を仮定した PERT スケジューリングによって求められた開始工程と終了工程の山積開始時刻と完了時刻を表す。PERT スケジューリングでは、ジョブの納期を守りつつ、各工程はできるだけ遅く開始し、できるだけ早く終了する。また、各工程を処理する初期機械は、各工程で加工時間が一番長い機械、各工程で加工時間が一番短い機械、および納期の早いジョブから順番に各機械の負荷ができるだけ均等となるような機械を選択する 3 種類が採用される。このスケジューリング・ロジックでは、機械での工程の処理の中断は許さないことを仮定する。すべてのジョブの原材料が生産ラインへ到着する時刻は決まっており、それ以前に処理を開始できない。また、機械は常に利用可能であるとは限らず、故障や点検

のために止まっている場合も考慮する。

スケジューリング問題の決定変数は、各工程を処理する機械 $m_{i,j}$ とその機械での開始時刻 $b_{i,j}$ である。スケジューリングによって、次のように定義される目的関数 J_{IT}、または J_{ET} を最小とする全ジョブの全工程の処理機械 $m_{i,j}$ と開始時刻 $b_{i,j}$ を決定する。

$$J_{IT} = Q\sum_i w_i \left(\sum_{r\in R_i} I_{i,r}^2 + \sum_{g\in G_i} T_{i,g}^2 \right)、Q = \frac{1}{\sum_i (|R_i| + |G_i|)} \tag{4}$$

$$J_{ET} = Q\sum_i w_i \left(\sum_{g\in G_i} E_{i,g}^2 + \sum_{g\in G_i} T_{i,g}^2 \right)、Q = \frac{1}{\sum_i |G_i|} \tag{5}$$

一般に納期遵守率の向上とリードタイムの短縮は生産管理上の重要な目標となる。前者は納期遅れのペナルティ T_i によって、そして後者は投入進みのペナルティ I_i によって作成するスケジュールに反映される。ここで、ジョブ i の工程 j を時刻 τ に機械 $m_{i,j}$ で処理中の場合は 1、それ以外は 0 となる $\delta_{i,j,m_{i,j},\tau}$ を導入して、機械制約は次のように書ける。

$$\sum_{i,j} \delta_{i,j,m_{i,j},\tau} \le A_{m,\tau}、\tau = 1,2,...,H \tag{6}$$

ここで $A_{m,\tau}$ は、時刻 τ に機械 m が稼動していれば 1、それ以外は 0 の値を取る。ジョブ i の工程 j の先行工程の集合を $P_{i,j}$ とすると、先行関係制約は以下のとおり。

$$c_{i,k} + s_{i,k} \le c_{i,j} - t_{i,j,m_{i,j}}、k \in P_{i,j} \tag{7}$$

また、到着時刻制約と処理時間要件は次の通り。

$$a_{i,r} \le c_{i,r} - t_{i,r,m_{i,r}}、r \in R_i \tag{8}$$

$$c_{i,j} - b_{i,j} + 1 = t_{i,j} \tag{9}$$

2　ラグランジュ緩和法

このスケジューリング・ロジックでは、ラグランジュ乗数を使って機械制約を緩和し、問題をジョブレベルの子問題に分解する [1]。工程レベルへの分解は、元の問題と緩和問題の乖離が大きい、またラグランジュ乗数が多くなるためにそれらの更新に時間がかかることが知られているので採用しない [2]。機械制約 (6) は、非負のラグランジュ乗数 $\lambda_{m,\tau} \ge 0$（$m = 1,...,M$、$\tau = 0,1,...,H$）を使って緩和される。機械制約の緩和によって以下の緩和問題が得られる。（以下では J_{IT} 型の目的関数の場合について述べる。）

$$R = \min_{m_{i,j},b_{i,j}} \left\{ J + \sum_{\tau=0}^{H}\sum_{i=1}^{N}\sum_{j=1}^{n_i} \lambda_{m_{i,j},\tau}\delta_{i,j,m_{i,j},\tau} - \sum_{m}^{M}\sum_{\tau=0}^{H} \lambda_{m,\tau}A_{m,\tau} \right\} \tag{10}$$

ジョブレベルに分解された子問題は次のように書ける。

$$L_i = \min_{m_{i,j},b_{i,j}} \left\{ w_i Q \left(\sum_{r\in R_i} I_{i,r}^2 + \sum_{g\in G_i} T_{i,g}^2 \right) + \sum_{\tau=0}^{H}\sum_{j=1}^{n_i} \lambda_{m_{i,j},\tau}\delta_{i,j,m_{i,j},\tau} \right\} \tag{11}$$

ここで、ジョブにおける技術的順序の制約は子問題に含まれたままであることに注意が必要である。

緩和問題の双対問題は次の式で与えられる。

$$\begin{aligned} L &= \max_{\lambda_{m,\tau}} \left\{ -\sum_{m=1}^{M}\sum_{\tau=0}^{H} \lambda_{m,\tau} A_{m,\tau} \right. \\ &\quad \left. + \sum_{i=1}^{N} \min_{m_{i,j},b_{i,j}} \left(w_i Q \left(\sum_{r\in R_i} I_{i,r}^2 + \sum_{g\in G_i} T_{i,g}^2 \right) + \sum_{j=1}^{n_i} \lambda_{m_{i,j},\tau} \delta_{i,j,m_{i,j},\tau} \right) \right\} \\ &= \max_{\lambda_{m,\tau}} \left\{ -\sum_{m=1}^{M}\sum_{\tau=0}^{H} \lambda_{m,\tau} A_{m,\tau} + \sum_{i=1}^{N} L_i \right\} \end{aligned} \tag{12}$$

双対問題でもジョブは独立であるため、双対問題のジョブの和の中に子問題 (11) が現れる。ラグランジュ緩和法を使って近似最適解を得る手順には、主に (11) の子問題を解く、(12) の双対問題を解く、実行可能解を構成するの３つのステップがある。ラグランジュ緩和法のアルゴリズムの効率性は、分解された子問題を解くアルゴリズムに依存する。サクラスケジューラに組み込まれているラグランジュ緩和法を使ったスケジューリング・ロジックでは、子問題を解くのに動的計画法を使う [3]。双対問題は、ジョブ毎の子問題を解く度にラグランジュ乗数 $\lambda_{m,\tau}$ を更新する代理劣勾配法によって解く。劣勾配法と比較して、代理劣勾配法は計算が高速であることが知られている [4]。また、実行可能解を構成するのには、子問題を解くことによって求めた各ジョブの各工程の開始時刻 $b_{i,j}$ を使ったリストスケジューリングが適用される。

3　子問題を解く

子問題 (11) の最適解は動的計画法を使って効率的に得られる [3]。これ以降では簡単のために、ジョブ i の添え字を省略する。例えば、ジョブ i の工程 j の完了時刻を $c_{i,j} \to c_j$ と書く。任意の $\tau = 1, ...H$、$1 \le j \le n$ と $\mu \in \mathcal{M}_j$ に対し、$\pi_{\tau,\mu}^j = \sum_{\theta=\tau-t_{j,\mu}}^{\tau-1} \lambda_{\mu,\theta} + \eta_{\tau,j,\mu}$ とする。ここで、$\eta_{\tau,j,\mu}$ は次式で定義される。

$$\eta_{\tau,j,\mu} = \begin{cases} 0 & j \notin R_i, G_i、 \\ Q \max\{0, r_j - \tau\}^2 & j \in R_i、 \\ Q \max\{0, \tau + t_{j,\mu} - d_j\}^2 & j \in G_i \end{cases} \tag{13}$$

それゆえ、$\pi_{\tau,\mu}^j$ は工程 j を機械 μ で τ に処理を開始する、または工程 j を機械 μ で $\tau + t_{j,\mu}$ に処理を完了させるコストであると言える。$\pi_{\tau,\mu}^j$ を使って、子問題 (11) は次のように書ける。

$$\min_{b_j,m_j} \sum_{j=1}^{n} \pi_{b_j,m_j}^j \tag{14}$$

今、パラメーター $1 \le k \le n$、$0 \le x \le H$ そして $u \in \mathcal{M}_k$ を使って $S_k(x,u)$ を次のように定義する。

$$S_k(x,u) = \min_{b_j,m_j} \sum_{j\in P_k} \pi_{b_j,m_j}^j \tag{15}$$

$S_k(x,u)$ は先行関係制約（7）を満足する必要があり、$c_k = x, m_k = u$ である。ここで、P_k は工程 k の先行工程の集合である。$S_k(x,u)$ の最適値 $f_k(x,u)$ は次の式で与えられる。

$$\begin{aligned} f_k(x,u) &= \pi_{x,u}^k、|P_k| = 0、 \\ f_k(x,u) &= \pi_{x,u}^k + \sum_{j\in P_k} \min_{v\in\mathcal{M}_j} g_j(x - t_{k,u} - s_k, v) \end{aligned} \tag{16}$$

ここで、

$$g_j(y,v) = \min_{0 \le z \le y} f_j(z,v) \tag{17}$$

である。(16) は動的計画法の境界条件と再帰式を表す。子問題 (14) の最適解は $g_n(H,\mu)$ によって与えられる。

4　双対問題を解く

決定変数が $\lambda_{m,\tau}$ である双対問題 (12) の解を得るために代理劣勾配法を使う。一般的な劣勾配法と代理劣勾配法の違いは、前者ではすべてのジョブの子問題を解いてから $\lambda_{m,\tau}$ を更新するのに対して、後者は 1 つのジョブの子問題を解いたら他のジョブの $b_{i,j}$ は変わらないと考えて $\lambda_{m,\tau}$ を更新する点にある。代理劣勾配法を使うことで、双対問題の解の収束が早くなり計算量が減らせることが知られている [4]。n 回更新した $\lambda_{m,\tau}$ を $\lambda^n_{m,\tau}$ と書くと、ラグランジュ乗数 $\lambda^n_{m,\tau}$ はステップサイズ α^n と λ に関する双対関数 L の劣勾配 $g_{m,\tau}(\lambda^n)$ に応じて変化する。

$$\lambda^{n+1}_{m,\tau} = \lambda^n_{m,\tau} + \alpha^n g_{m,\tau}(\lambda^n) \tag{18}$$

劣勾配 $g_{m,\tau}(\lambda^n)$ は、機械制約 (6) に従って次のように書ける。

$$g_{m,\tau}(\lambda) = \sum_{i,j} \delta_{i,j,m,\tau} - A_{m,\tau} \tag{19}$$

ステップサイズ α^n は、n 回目の双対関数の値 L^n、最適値の見積もり $\overline{L}$、劣勾配の内積、そしてパラメータ a の関数である。

$$\alpha^n = a\frac{\overline{L} - L^n}{g(\lambda^n)^T g(\lambda^n)}、0 < a < 2 \tag{20}$$

代理劣勾配アルゴリズムは、一定の更新回数経っても L^n が改善されなかったら終了する。

5　実行可能解を構成する

離散的な決定変数が含まれていること、および終了条件のために、双対問題 (12) の解は一般的に実行不可能である。すなわち、機械制約 (6) を破っている。ここで、子問題を解く手法の特徴から、技術的順序制約 (7) と処理時間要件 (9) は常に満たされている。実行可能解を構成するために、リストスケジューリングの概念に基づく発見的な手法を使う。双対問題を解く過程で各ジョブの子問題を解く際に、各工程を処理する機械 $m^*_{i,j}$ とその機械での開始時刻 $b^*_{i,j}$ が決定している。リストは、全ジョブの全工程の $b^*_{i,j}$ の昇順で各工程を並べることによって生成される。工程は、技術的順序制約、機械制約および処理時間要件を満たしつつ、このリストの順番に機械 $m^*_{i,j}$ に割り当てられる。

サクラスケジューラに組み込まれているラグランジュ緩和法を使ったスケジューリング・ロジックでは、実行可能解を構成する方法としてこのリストスケジューリングに基づく 3 つの手法が使われる。1 つは、機械制約が破られたときに貪欲法を使ってどのジョブのどの工程を先に機械に割り当てるかを決める手法である。次は、機械制約が破られたときに開始時刻 $b^*_{i,j}$ を基点として機械に割り当て可能な時刻を前方および後方探索し、見つかった両方向の時刻のうち $b^*_{i,j}$ に近い方を採用する手法である。最後は、他のどの作業の処理開始時刻にも影響を与えないで他の作業を飛び越してより早く着手できる作業を順次前に移動することによって、他の作業の着手を遅らせるのでなければもはやどの作業も前に移動できないように工程を割り付ける方法である。

5.1　貪欲法を使った実行可能解の構成

目的関数が納期遅れ（tardiness）だけの場合、リストの順番に時刻の前詰めで機械に割り当てていき、時刻 k において機械制約が破られたら貪欲法を使ってどの工程を時刻 k に割り当てるかを決定するのが一般的である [1]。しかしながら投入進みも目的関数に含まれる場合、このアルゴリズムはそのままでは適用できない。サクラスケジューラのラグランジュ緩和法を使ったスケジューリング・ロジックでは、投入進みペナルティは子問題 (14) の解 $b^*_{i,j}$ のときの値以上に悪くしないという制約の下で各ジョブの各工程をリストの順番に時刻 $b^*_{i,j}$ に機械に割り当てていき、時刻 k において機械制約が破られたら同じく貪欲法を使ってどの工程を後ろにずらすのかを決定する。実行可能解を構成する詳細な手順は次のとおり。

- ステップ 1：各機械の利用可能機械台数 $A_{m,\tau} = 0$ か 1 を機械台数に設定する。次に、全工程を開始時刻 $b^*_{i,j}$ の昇順で並べて $\tau = 0$ とする。この工程の集合を S とする。
- ステップ 2：$b^*_{i,j} = \tau$ となる工程の集合 E_τ を抽出する。次いで、E_τ の工程を山積開始時刻 $r_{i,j}$ と緩和解 $b^*_{i,j}$ との差が大きい順に並べ替える。
- ステップ 3：$p \in E_\tau$、$p = 1, ..., |E_\tau|$ と p を定義し、$p = 1$ とする。
- ステップ 4：工程 p が $b^*_{i,p}$、...、$b^*_{i,p} + t_{i,p}$ の間 $A_{m_{i,p},\tau} > 0$ なら工程 p を $b^*_{i,p}$ に割り付ける。$b^*_{i,p}$、...、$b^*_{i,p} + t_{i,p}$ の間 $A_{m_{i,p},\tau} = A_{m_{i,p},\tau} - 1$ とし、$S = S - \{i, p\}$ とする。さもなければ $b^*_{i,p} = b^*_{i,p} + 1$ とし、後続工程を技術的順序を破らないように後ろにずらす。ステップ 5 に行く。
- ステップ 5：$p = |E_\tau|$ ならステップ 6 に行く。そうでないなら $p = p + 1$ としてステップ 4 に行く。
- ステップ 6：$|S| = 0$ なら処理終了。さもなくば、$\tau = \tau + 1$ とし、残っている工程の集合 S を開始時刻の昇順で並べなおす。ステップ 2 に行く。

5.2　探索法を使った実行可能解の構成

貪欲法を使った実行可能解の構成は、そのアルゴリズムの特性から、各時刻に対するジョブの密度（負荷の大きさ）に強く依存し、密度が高い場合に計算時間がかなり多くなる。それゆえジョブ数が数百の小規模な問題に対してはこのアルゴリズムは有効であるが、ジョブ数が数千の大規模な問題に対しては、より高速な実行可能解の構成方法が求められる。サクラスケジューラのスケジューリング・ロジックで採用されている簡単かつ高速に実行可能解を構成する詳細な手順は次のとおり。

- ステップ 1：各機械の利用可能機械台数 $A_{m,\tau} = 0$ か 1 を機械台数に設定する。次に、全工程を開始時刻 $b^*_{i,j}$ の昇順で並べ、この工程の集合を S とする。$p \in S$、$p = 1, ..., |S|$ と p を定義し、$p = 1$ とする。
- ステップ 2：工程 p が $b^*_{i,p}$、...、$b^*_{i,p} + t_{i,p}$ の間 $A_{m_{i,p},\tau} > 0$ なら工程 p を $b^*_{i,p}$ に割り付け、ステップ 3 に行く。さもなければ ステップ 4 に行く。
- ステップ 3：$b^*_{i,p}$、...、$b^*_{i,p} + t_{i,p}$ の間 $A_{m_{i,p},\tau} = A_{m_{i,p},\tau} - 1$ とし、$p = p + 1$ としてステップ 2 に行く。

- ステップ 4：$k = b^*_{i,p} + 1$ とする。
- ステップ 5：工程 p が k、...、$k + t_{i,p}$ の間 $A_{m_{i,p},\tau} > 0$ なら $K = k$ としてステップ 6 に行く。さもなければ $k = k + 1$ としてステップ 5 を繰り返す。
- ステップ 6：$l = b^*_{i,p} - 1$ とする。
- ステップ 7：工程 p が l、...、$l + t_{i,p}$ の間 $A_{m_{i,p},\tau} > 0$ なら $L = l$ としてステップ 8 に行く。また、$l < 0$ の場合、もしくは $|b^*_{i,p} - l| \geq |b^*_{i,p} - K|$ の場合もステップ 8 に行く。さもなければ $l = l - 1$ としてステップ 7 を繰り返す。
- ステップ 8：$|b^*_{i,p} - L| \geq |b^*_{i,p} - K|$ なら工程 p を時刻 K に割り付ける。K、...、$K + t_{i,p}$ の間 $A_{m_{i,p},\tau} = A_{m_{i,p},\tau}$ とする。さもなければ時刻 L に割り付ける。L、...、$L + t_{i,p}$ の間 $A_{m_{i,p},\tau} = A_{m_{i,p},\tau}$ とする。$p = |S|$ なら処理終了。さもなければ $p = p + 1$ としてステップ 2 に行く。

5.3　Giffler & Thompson 法を使った実行可能解の構成

実行可能スケジュールの集合の中でアクティブ・スケジュールの集合の中に最適なスケジュールが存在するは良く知られている [6]。ここで、アクティブ・スケジュールとは、各機械上での作業の処理順序を変更してもよいとして、他の作業の開始時刻を遅らせることなしに、いずれの作業も処理開始を早めることができないスケジュールである。Giffler & Thomspon 法によって、このアクティブ・スケジュールが構成可能である [5]。実行可能解を構成する詳細な手順は次のとおり。

- ステップ 1：各ジョブの開始工程の集合 S を抽出する。
- ステップ 2：集合 S から完了時刻のもっとも早い工程 $p \in S$ を選択する。
- ステップ 3：工程 p と同じ機械で処理される工程の集合 B を抽出する。
- ステップ 4：集合 B から工程 p の完了時刻以降に開始予定の工程を削除する。
- ステップ 5：集合 B から割り付け順序の最も早い工程 p^* を選択する。
- ステップ 6：p^* の開始時刻を探索法と同じロジックを使って決定し、$S = S - \{p^*\}$ とする。
- ステップ 7：p^* の後続工程を集合 S に加える。
- ステップ 8：$|S| = 0$ なら処理終了。さもなければステップ 2 に行く。

6　各スケジューリング・ロジック

サクラスケジューラに組み込まれているラグランジュ緩和法を使った各スケジューリング・ロジックで採用されている実行可能解の構成方法は次のとおり。

- LR_G_Scheduler：実行可能解の構成に貪欲法のみを使う。
- LR_S_Scheduler：実行可能解の構成に探索法のみを使う。

- LR_H_Scheduler：双対問題を解いている間は探索法を使い、全工程の処理機械 $m^*_{i,j}$ と開始時刻 $b^*_{i,j}$ が求まった段階で貪欲法によって実行可能解を構成し直す。ただし、探索法による実行可能解の方が最適度が良い場合は、探索法よる実行可能解を近似最適解とする。
- LR_T_Scheduler：実行可能解の構成に Giffler & Thompson 法のみを使う。

参考文献

[1] D.J. Hoitomt, P.B. Luh, E. Max and K.R. Pattipati, Shceduling Jobs with Simple Precedence Constraints on Parallel Machines, *IEEE Control Systems Magazine*, 1990.

[2] D.J. Hoitomt, P.B. Luh and K.R. Pattipati, A Practical Approach to Job-Shop Scheduling Problems, *IEEE Transaction on Robotics and Automation*, Vol.9, No.1, 1993.

[3] H. Chen, C. Chu and J.M. Proth, A More Efficient Lagrangian Relaxation Approach to Job-Shop Scheduling Problems, *IEEE International Conference on Robotics and Automation*, pp.496-501, 1995.

[4] X. Zhoa, P.B. Luh and J. Wang, The Surrogate Gradient Algorithm for Lagrangian Relaxation Method, *IEEE Conference on Decision and Control*, 1997.

[5] B. Giffler and G. Thompson, Algorithms for Solving Production Scheduling Problems, *Operations Resarch*, Vol.8 pp.487-503, 1960

[6] C. Bierwirth and D. Mattfeld, Production Scheduling and Rescheduling with Genetic Algorithms, *Evolutionary Computation*, Vol.7 pp.1-17, 1999

