JPS 할당규칙의 Transitivity 증명을 통한 총납기지연 최소화

전태준(전남대 산업공학과) · 박성호(담양대 전산정보통신공학부)

tjjeon@chonnam.chonnam.ac.kr
shpark@damyang.damyang.ac.kr

요약 총납기지연 최소화 문제에서 많은 할당규칙들이 개발되어져 왔는데, 대부분의 개발 배경이 모든 Job이 동일한 시작시간을 갖는 Single Machine을 대상으로 하였기 때문에 이 할당규칙들을 Job의 시작시간이 다르게 주어지는 문제에 그대로 적용했을 때는 만족할만한 결과를 얻기가 쉽지 않다.
따라서 본 연구에서는 다른 시작시간을 갖는 Single Machine문제에서 총납기지연을 최소화하기 위하여 먼저, 새로운 작업쌍 순서(Job Pair Sequencing) 할당규칙을 제시한다. 제시되는 할당규칙은 Job들간의 Sequence를 쌍으로 비교하여 총납기지연 측면에서 선행이 바람직한 Job을 선택한다. 그리고 제시된 할당규칙에 대하여 다수개의 Job에 적용 시 선택된 Job들간에 Transitivity 가 성립됨을 증명함으로서 총납기지연을 최소화 시킬 수 있음을 증명하였다.

Ⅰ. 서론

납기에 관련된 스케쥴링 문제에서 Single Machine이 이론적인 연구대상으로 고려되는 이유는 Random한 Job 도착의 영향 또는 납기설정 방법론을 고려하지 않은 채 다양한 Sequencing 절차들의 직접적인 비교를 할 수 있으며, 또한 그 자체로도 중요한 연구문제가 된다는데 있다[7].

따라서 Single Machine 문제는 많은 연구자들에 의해서 연구되어져 왔는데, 먼저, Job의 Adjacent Pairwise Interchange 속성에 대한 개념을 기반으로 Baker[2]가 선행이 바람직한 Job을 선택하기 위한 조건식을 제시하였고, 이를 기반으로 Wilkerson과 Irwin[WI]에 의해 근사적 기법이 개발되었다[8]. 또 다른 초기 할당규칙으로 SPT(Shortest Processing Time)와 EDD(Earliest Due Date)가 제시되었으며, SPT와 EDD 규칙을 기반으로 Baker와 Bertrand[3]가 MDD(Modified Due Date) 규칙을 제안하였는데, 다른 규칙에 대한 MDD의 우수성은 수치적으로 입증되었다[3, 4].

최근에는 Panwaker, Smith와 Koulamas[6]가 새로운 규칙인 PSK를 제시하였고, 수치상으로 실험하여 최적해를 얻는데 있어서 타 규칙보다 상대적으로 우수함을 보였다. 또한 Holsenback과 Russell[5]은 Emmons의 강력한 Dominance 성질을 사용하는 새로운 NBR을 개발하였는데 총납기지연에 있어서 커다란 감소를 이루었다. 하지만 해는 대부분 작은 문제에 대하여 최적이고, 다른 문제에 대하여는 대부분 최적 근사해를 구할 뿐이었다[1].

본 논문에서는 먼저, 총납기지연을 최소화 시킬 수 있는 새로운 할당규칙인 JPS(Job Pair Sequencing)를 제시하고자 한다. 제시되는 할당규칙은 Single Machine에서 Job의 시작시간이 다르게 주어지는 경우를 대상으로 모든 스케쥴 대상 Job들을 두 개씩 쌍으로 순서관계를 비교하면서 최종적으로 선행이 바람직한 하나의 Job을 선택한다.

그러나 제시되는 할당규칙은 두 개 Job 간의 순서관계의 비교에서 선행이 바람직하지 않다고 판단된 Job을 그대로 스케쥴 대상 Job으로 고려할 것인지 아니면 제외시켜도 될 것인지에 대한 이론적인 증명이 함께 제시되어야 한다. 그렇지 않으면 기존의 일반적인 할당규칙들에 비하여 스케쥴 대상 Job들간에 요구되는 비교횟수가 급격하게 증가되는 문제점이 있고, 선행이 바람직한 Job을 선택하는 과정에서 순환이 발생되어 어떠한 Job도 선행이 바람직하다고 판단되지 못하는 경우도 발생될 수 있다.

따라서 본 논문에서는 이에 대한 이론적인 증명으로 제시되는 할당규칙의 적용 시 선택되는 Job 간에는 이행성(Transitivity)이 성립됨을 증명하고자 하였다. 이행성을 증명함으로서 제시되는 할당규칙 적용 시 경쟁에서 탈락된 Job은 스케쥴 고려대상에서 바로 제외시켜도 되며, 비교에서 최종적으로 선택된 Job이 가장 바람직한 선택임을 보일 수 있고 총납기지연을 최소화 시킬 수 있음을 보였다.

본 논문의 내용은 모두 5장으로 구성되어 있다. 1장은 서론으로 논문의 연구배경과 연구내용 등을 기술하였다. 2장에서는 JPS(Job Pair Sequencing) 할당규칙의 개념을 설명하였고, 3장에서 제시된 할당규칙 적용 시 이행성(Transitivity)이 성립되는 증명과정을 보였다. 4장에서는 결론부분으로서 연구내용을 종합하였다.

II. JPS 할당규칙

1. JPS 할당규칙

본 논문에서 새롭게 제시되는 할당규칙의 특징은 여러 스케쥴 대상 Job들 중에서 총납기지연 측면에서 선행이 바람직한 하나의 Job을 선택하기 위하여, 두 개 Job씩 쌍으로 비교해 나가는 개념을 이용한다는 점이다. 따라서 이 할당규칙을 본 논문에서는 작업쌍 순서(Job-Pair Sequencing) 할당규칙이라고 명명한다(이후 JPS 할당규칙이라고 한다).

JPS 할당규칙에 대한 개념을 설명하기 위하여 임의로 두 개의 Job i와 j를 고려하자. 두 개의 Job에서 가능한 순서관계는 Job i가 선행되는 경우의 Sequence S와 그 반대의 경우인 S′가 고려될 수 있으며, 이 때 각 순서관계에서의 평가함수 값의 비교를 통하여 선행이 바람직한 Job을 판단한다. 먼저, 각 순서관계에서의 총납기지연 값은 다음 (식 2.1)과 같이 계산된다.

$$
\begin{aligned}
T_{ij} &= T_i(s) + T_j(s) \\
&= \max\{(s_i + p_i - d_i), 0\} + \max\{(s_i + p_i + p_j - d_j), 0\}
\end{aligned}
$$

(식 2.1)

$$
\begin{aligned}
T_{ji} &= T_j(s') + T_i(s') \\
&= \max\{(s_j + p_j - d_j), 0\} + \max\{(s_j + p_j + p_i - d_i), 0\}
\end{aligned}
$$

여기서 T_{ij} 와 T_{ji} 값을 비교하여 더 작은 값을 갖는 순서관계에서의 선행 Job만을 선행이 바람직한 것으로 선택하는 것이 JPS 할당규칙이다. 본 논문에서는 Job i가 선행되는 것이 바람직한 관계임을 표현하기 위하여 $i \rightarrow j$ 기호를 사용하기로 하고, 그 반대의 경우는 $j \rightarrow i$ 기호를 사용하기로 한다. 이 기호를 사용하여 JPS 할당규칙에 따른 선행이 바람직한 순서관계는 다음과 같이 두 가지 경우로 구분된다.

ⓐ $T_{ij} < T_{ji} \Rightarrow i \rightarrow j$
ⓑ $T_{ij} \geq T_{ji} \Rightarrow j \rightarrow i$

2. 조건별 순서관계

각 Job들이 다른 시작시간을 갖는 경우에서는 (식 2.1)에서 선행이 바람직한 Job을 표현하는 조건식을 정리하기 위해서 많은 경우로 구분하여 살펴보아야 되고, 이러한 각 경우에서 조건식을 간략화하기가 쉽지 않다. 따라서 본 논문에서는 고려되는 많은 경우를 쉽게 표현하기 위한 방법의 하나로서 (식 2.1)의 각 Sequence에서 Job 들이 갖는 값의 형태를 기호를 사용하기로 가정한다. 이를 위해 납기지연 되지 않는(Earliness) 경우는 (○) 기호를, 납기지연(Tardiness) 되는 경우는 (□)기호를 사용하기로 정의한다.

먼저, (식 2.1)의 각 Sequence에서 Job들이 갖는 값의 형태에 따라 실제로 발생 가능한 모든 경우는 9가지 경우이며 이를 정의된 기호로 표현한 뒤, T_{ij}와 T_{ji} 의 크기 관계를 알 수 있는 경우에서는 선행이 바람직한 Job이 쉽게 결정된다. 그러나 그렇지 못한 경우에서는 각각의 경우에서 선행이 바람직한 Job이 결정될 수 있는 조건식, 즉 $T_{ij} < T_{ji}$ 또는 $T_{ij} \geq T_{ji}$ 가 될 수 있는 조건식이 구분되어 정리된다. 여기에 해당되는 경우는 4가지이며, 이에 대한 조건식이 유도되는 과정을 살펴보면 다음과 같다.

① $\begin{pmatrix} \bigcirc & \square \\ \bigcirc & \square \end{pmatrix}$

$$\begin{aligned} T_{ij} - T_{ji} &= (s_i + p_i + p_j - d_j) - (s_j + p_j + p_i - d_i) \\ &= (s_i + d_i) - (s_j + d_j) \end{aligned}$$

if $(s_i + d_i) < (s_j + d_j)$ 이면 $T_{ij} < T_{ji}$: $(i \rightarrow j)$

else $(s_i + d_i) \geq (s_j + d_j)$ 이면 $T_{ij} \geq T_{ji}$: $(j \rightarrow i)$

② $\begin{pmatrix} \bigcirc & \square \\ \square & \square \end{pmatrix}$

$$\begin{aligned} T_{ij} - T_{ji} &= (s_i + p_i + p_j - d_j) - (s_j + p_j - d_j) - (s_j + p_j + p_i - d_i) \\ &= (s_i + d_i) - (2s_j + p_j) \end{aligned}$$

if $(s_i + d_i) < (2s_j + p_j)$ 이면 $T_{ij} < T_{ji}$: $(i \rightarrow j)$

else $(s_i + d_i) \geq (2s_j + p_j)$ 이면 $T_{ij} \geq T_{ji}$: $(j \rightarrow i)$

③ $\begin{pmatrix} \square & \square \\ \bigcirc & \square \end{pmatrix}$

$$\begin{aligned} T_{ij} - T_{ji} &= (s_i + p_i - d_i) + (s_i + p_i + p_j - d_j) - (s_j + p_j + p_i - d_i) \\ &= (2s_i + p_i) - (s_j + d_j) \end{aligned}$$

if $(2s_i + p_i) < (s_j + d_j)$ 이면 $T_{ij} < T_{ji}$: $(i \rightarrow j)$

else $(2s_i + p_i) \geq (s_j + d_j)$ 이면 $T_{ij} \geq T_{ji}$: $(j \rightarrow i)$

④ $\begin{pmatrix} \square & \square \\ \square & \square \end{pmatrix}$

$$\begin{aligned} T_{ij} - T_{ji} &= (s_i + p_i - d_i) + (s_i + p_i + p_j - d_j) - (s_j + p_j - d_j) - (s_j + p_j + p_i - d_i) \\ &= (2s_i + p_i) - (2s_j + p_j) \end{aligned}$$

if $(2s_i + p_i) < (2s_j + p_j)$ 이면 $T_{ij} < T_{ji}$: $(i \rightarrow j)$

else $(2s_i + p_i) \geq (2s_j + p_j)$ 이면 $T_{ij} \geq T_{ji}$: $(j \rightarrow i)$

III. Transitivity 증명

JPS 할당규칙에 의해 두 개 Job 간에 바람직한 순서관계는 조건별로 쉽게 구분되어 진다. 하지만 JPS 할당규칙이 다수개의 Job에 대하여 적용된다고 했을 때, 두 개 Job 중에서 선택되지 않은 Job의 처리 문제 때문에 절차는 그리 간단하지 않게 된다. 즉, JPS 할당규칙을 적용했을 때 선택되지 않은 Job을 현재 스케쥴 고려 대상에서 제외시킬 수 있는 이론적인 증명이 이루어지지 않는다면, n 개의 Job에서 요구되어지는 Computational Complexity가 $O(n^2)$으로 부담이 되기 때문이다. 그렇다고 선택되지 않은 Job을 현재 스케쥴 고려 대상에서 그냥 제외했을 경우에는 n 개의 Job 중에서 최종 선택된 Job이 나머지 (n-1) 개의 Job보다 항상 바람직한 선택이었는지 확신할 수 없다.

본 논문에서는 이에 대한 이론적인 증명으로 JPS 할당규칙 적용 시 Job 들간에 Transitivity가 만족됨을 보이고자 한다. 그 결과로서 요구되는 Computational Complexity를 O(n)으로 감소시킬 수 있고, JPS 할당규칙만으로도 총납기지연을 최소화 할 수 있음을 보이고자 한다.

그러면 n 개의 Job간의 JPS 할당규칙 적용 시 Transitivity가 만족됨을 보이기 위하여 임의로 세 개의 Job i, j 그리고 k를 가정하기로 한다. 이 때 세 개의 Job 들이 다음과 같은 (Theorem. 1)을 만족한다면 JPS 할당규칙은 총납기지연을 최소화할 수 있다.

(Theorem. 1)

세 개의 Job i, j 그리고 k에 대하여 JPS 할당규칙을 적용했을 때 $i \rightarrow j$ 이고 $j \rightarrow k$ 이면 $i \rightarrow k$를 의미한다면 Transitivity가 만족된다고 할 수 있다.

(Theorem 1)의 증명을 위하여 먼저, $i \rightarrow j$가 되는 조건식들만을 살펴보면 7가지 경우가 존재하고, 이것을 정리하면 다음 아래 [표-1]과 같다.

[표-1] $i \rightarrow j$ 순서관계에서의 조건식

	Condition	
$(i \rightarrow j)_{①}$	○+○ ○+○	$2s_i + p_i < s_j + d_j$
$(i \rightarrow j)_{②}$	○+○ ○+□	
$(i \rightarrow j)_{③}$	□+○ ○+□	
$(i \rightarrow j)_{④}$	○+□ ○+□	$s_i + d_i < s_j + d_j$
$(i \rightarrow j)_{⑤}$	○+□ □+□	$s_i + d_i < 2s_j + p_j$
$(i \rightarrow j)_{⑥}$	□+□ ○+□	$2s_i + p_i < s_j + d_j$
$(i \rightarrow j)_{⑦}$	□+□ □+□	$2s_i + p_i < 2s_j + p_j$

결국 JPS 할당규칙에 대한 Transitivity가 만족됨을 보이기 위해서는, 위와 같이 $i \rightarrow j$가 되는

7가지 경우의 조건들과 $j \rightarrow k$ 가 되는 7가지 경우의 조건들이 결합되어 나타날 수 있는 모든 Case를 고려해야 한다. 그런데, Job j가 동일한 Sequence 위치에서 상반된 납기지연 형태를 보이는 Case는 Transitivity 증명과정에서 발생 불가능한 경우이기 때문에 이를 제외한 나머지에 대한 증명만이 실제적으로 요구된다고 볼 수 있다.

한편, Transitivity 증명을 위한 각각의 Case에서 쌍으로 비교되는 Job이 다르더라도 Job i, j 그리고 k에 대한 선행위치에서의 납기지연 형태는 동일하게 나타난다. 그래서 증명을 위해서 비교되는 Job i 와 k 가 선행위치에서 갖게되는 납기지연 형태에 대한 정보는 미리 알 수 있다.

따라서 Transitivity 증명이 요구되는 각각의 Case를 Job i 와 k 가 선행위치에서 갖는 납기지연 형태에 따라 편의상 4가지로 그룹화가 가능하고, 각 Job이 후행 할 때 나타날 수 있는 납기지연 형태에 따라 구분하여 각각에 대한 세부적인 증명이 요구된다. 그러면 각 Group 별로 세부적인 (Theorem. 1)에 대한 증명과정을 살펴보자.

(1) Group I

(예) $Condition\,(i \rightarrow j)_{③}$ 와 $Condition\,(j \rightarrow k)_{⑥}$ 이 결합되는 Case

(가정 i) $s_i + p_i + p_k < d_k$ 이고 $s_k + p_k + p_i < d_i$ 인 관계에서는 어떤 Sequence에서도 납기지연이 발생되지 않기 때문에 이 때는 $Condition(i \rightarrow k)_{①}$ 로 (Theorem. 1)이 증명된다.

(가정 ii) $s_i + p_i + p_k > d_k$ 이고 $s_k + p_k + p_i < d_i$ 인 관계에서는 먼저, 두 식을 i 와 k 에 대하여 다시 정리한 뒤에 서로 결합하면, $s_i + d_i > s_k + d_k$ 관계가 된다고 볼 수 있다. 그런데, 이 식은 주어진 Condition에서 $s_i + d_i < 2s_j + p_j < s_k + d_k$, 즉 $s_i + d_i < s_k + d_k$ 관계를 얻을 수 있기 때문에 이는 가정에서 얻어진 관계와 서로 모순된다. 따라서 발생되지 않는 경우이다.

(가정 iii) $s_i + p_i + p_k < d_k$ 이고 $s_k + p_k + p_i > d_i$ 인 관계에서는 쉽게 $Condition(i \rightarrow k)_{②}$ 로 (Theorem. 1)이 증명된다.

(가정 iv) $s_i + p_i + p_k > d_k$ 이고 $s_k + p_k + p_i > d_i$ 인 관계에서는 주어진 Condition에서 $s_i + d_i < s_k + d_k$ 관계가 얻어지기 때문에 $Condition(i \rightarrow k)_{④}$로 (Theorem. 1)이 증명된다.

(2) Group II

(예) $Condition\,(i \rightarrow j)_{⑤}$ 와 $Condition\,(j \rightarrow k)_{⑦}$ 이 결합되는 Case

(가정 i) $s_k + p_k + p_i < d_i$ 인 관계에서는, 먼저 양쪽에 s_i를 더하면 $s_i + s_k + p_k + p_i < s_i + d_i$ 와 같이 식을 변화시킬 수 있다. 그리고 이 식을 Condition으로부터 유도되는 $s_i + d_i < 2s_k + p_k$ 식과 함께 결합하면, $s_i + p_i < s_k$ 관계가 된다.

그런데, 이 식은 앞에서 두 개 작업이 비교되기 위해 전제한 Active 조건에 위배된다. 즉, 두 개의 Job i 와 k 가 비교되지 않는 비경쟁관계에 있기 때문에 Acrive 스케쥴을 가정했을 때 발생되지 않는다.

(가정 ii) $s_k + p_k + p_i > d_i$ 인 관계에서는 주어진 Condition에서 $s_i + d_i < 2s_j + p_j < 2s_k + p_k$ 즉, $s_i + d_i < 2s_k + p_k$ 관계가 유도되기 때문에 이는 $Condition(i \rightarrow k)_{⑤}$로 (Theorem. 1)이 증명된다.

(3) Group III

(예) *Condition* $(i \to j)_{⑦}$ 과 *Condition* $(j \to k)_{⑥}$ 이 결합되는 Case

(가정 i) $s_i + p_i + p_k < d_k$ 관계에서는 쉽게 조건식 *Condition*$(i \to k)_{③}$ 으로 (Theorem. 1)이 증명됨을 알 수 있다.

(가정 ii) $s_i + p_i + p_k > d_k$ 관계에서는 주어진 Condition에서 $2s_i + p_i < 2s_j + p_i < s_k + d_k$ 즉, $2s_i + p_i < s_k + d_k$ 관계가 유도되므로 이는 *Condition*$(i \to k)_{⑥}$ 으로 (Theorem. 1)이 증명된다.

(4) Group Ⅳ

(예) *Condition* $(i \to j)_{⑦}$ 과 *Condition* $(j \to k)_{⑦}$ 이 결합되는 Case

여기서는 바로 주어진 Condition으로부터 $2s_i + p_i < 2s_j + p_j < 2s_k + p_k$ 관계, 즉 $2s_i + p_i < 2s_k + p_k$ 에 의해 *Condition*$(i \to k)_{⑦}$ 로 (Theorem. 1)이 증명된다.

위에서 고려된 결합 Case 이외의 모든 Case에 대한 증명과정도 거의 유사하다. 따라서 모든 Case에 대한 증명을 수행한 결과를 [표-2]에 정리하였다. 그 결과 고려되는 모든 Case에서 (Theorem. 1)이 증명됨을 알 수 있고, 즉 JPS 할당규칙 적용 시 Transitivity 가 만족됨을 알 수 있다.

[표-2] Theorem 1 증명

$(j \to k)$ \ $(i \to j)$	$(j \to k)_{①}$	$(j \to k)_{②}$	$(j \to k)_{③}$	$(j \to k)_{④}$	$(j \to k)_{⑤}$	$(j \to k)_{⑥}$	$(j \to k)_{⑦}$
$(i \to j)_{①}$	$(ik)_{①}$ or $(ik)_{②}$ or $(ik)_{④}$	$(ik)_{①}$ or $(ik)_{②}$ or $(ik)_{④}$	×	$(ik)_{①}$ or $(ik)_{②}$ or $(ik)_{④}$	$(ik)_{⑤}$	×	×
$(i \to j)_{②}$	$(ik)_{①}$ or $(ik)_{②}$ or $(ik)_{④}$	$(ik)_{①}$ or $(ik)_{②}$ or $(ik)_{④}$	×	$(ik)_{①}$ or $(ik)_{②}$ or $(ik)_{④}$	$(ik)_{⑤}$	×	×
$(i \to j)_{③}$	$(ik)_{③}$ or $(ik)_{⑥}$	$(ik)_{③}$ or $(ik)_{⑥}$	×	$(ik)_{③}$ or $(ik)_{⑥}$	$(ik)_{⑦}$	×	×
$(i \to j)_{④}$	$(ik)_{①}$ or $(ik)_{②}$ or $(ik)_{④}$	$(ik)_{①}$ or $(ik)_{②}$ or $(ik)_{④}$	×	$(ik)_{①}$ or $(ik)_{②}$ or $(ik)_{④}$	$(ik)_{⑤}$	×	×
$(i \to j)_{⑤}$	×	×	$(ik)_{①}$ or $(ik)_{②}$ or $(ik)_{④}$	×	×	$(ik)_{①}$ or $(ik)_{②}$ or $(ik)_{④}$	$(ik)_{⑤}$
$(i \to j)_{⑥}$	$(ik)_{③}$ or $(ik)_{⑥}$	$(ik)_{③}$ or $(ik)_{⑥}$	×	$(ik)_{③}$ or $(ik)_{⑥}$	$(ik)_{⑦}$	×	×
$(i \to j)_{⑦}$	×	×	$(ik)_{③}$ or $(ik)_{⑥}$	×	×	$(ik)_{③}$ or $(ik)_{⑥}$	$(ik)_{⑦}$

Ⅳ. 결론

시작시간이 다르게 주어지는 Single Machine 문제를 대상으로 총납기지연을 최소화할 수 있는 새로운 JPS 할당규칙을 제시하였고, JPS 할당규칙이 총납기지연을 최소화 할 수 있음을 이론적으

로 보이기 위하여 선택된 Job간에 이행성(Transitivity)이 성립됨을 발생 가능한 모든 경우에 대하여 증명하였다. 더불어 이행성(Transitivity)이 증명됨에 따라 요구되는 Computational Complexity도 $O(n^2)$에서 $O(n)$으로 감소시킬 수 있었다.

참 고 문 헌

[1] Alidaee, B. and Gopalan, S., "A Note on the Equivalence of Two Heuristics to Minimize Total Tardiness", *EJOR 96*, pp514-517, 1997.

[2] Baker, K. R., *Elements of Sequencing and Scheduling*, 1998.

[3] Baker, K. R. and Bertrand, J. W., "A Dynamic Priority Rule for Scheduling Against Due-dates", *Journal of Operations Management 3*, pp37-42, 1982.

[4] Baker, K. R. and Kanet, J. J., "Job Shop Scheduling with Modified Due-dates", *Journal of Operations Management 4*, pp11-22, 1983.

[7] Holsenback, J. E., "A Pseudopolynomial Algorithm for Sequencing on One Machine to Minimize Total Tardiness", *Journal of the Operational Research Society 43*, pp53-62, 1992.

[6] Panwalkar, S. S., Simth, M. L. and Koulamas, C. P., "A Heuristic for the Single Machine Tardiness Problem", *EJOR 70*, pp304-310, 1993.

[7] Russell, R. M. and Holsenback, J. E., "Evaluation of Leading Heuristic for the Single Machine Tardiness Problem", *EJOR 96*, pp538-545, 1997.

[8] Wilkerson, J. and Irwin, J. D., "An Improved Algorithm for Scheduling Independent Tasks", *AIIE Transactions 3*, pp333-342, 1971.