スケジュール問題に対する有効な交叉法について

若宮　利治・片山　謙吾*・成久　洋之*

岡山理科大学大学院工学研究科修士課程情報工学専攻

*岡山理科大学工学部情報工学科

(1998年10月5日　受理)

1．はじめに

近年，システムの大規模化に伴い効率的な発見的手法の重要性が増してきている。そのような手法の一つとして，生物の進化過程を模倣した遺伝的アルゴリズム(Genetic Algorithm, GA)が注目されている。GAは生物進化現象に潜む情報多様化，環境適応，様々な戦略などにみられる問題解決能力などの進化プロセスを，情報処理モデル化して自然の問題解決機構を理解し，利用したものである。GAは様々な問題に適用され実用的にも成果をあげている。本研究ではスケジュール問題として有名なジョブショップスケジューリング問題（Job-shop Scheduling Problem：JSP，以下JSPと記す）を取り上げ，順序型交叉と位置型交叉の2つの交叉法を用いて比較検討を行う。

2．遺伝的アルゴリズム

遺伝的アルゴリズム（GA）は生物進化の原理に着想を得たアルゴリズムであり，最適化の一手法として考えることができる。GAを最適化問題に適用する場合，各解候補は，染色体と呼ばれる有限長文字列に符号化しなければならない。染色体を構成する文字列を遺伝子，染色体の集合を集団，集団に属する染色体数を集団サイズという。GAの特徴は，個体と呼ばれる複数の解候補を持ち，その各個体に対して適応度関数を用いて，選択，交叉，突然変異と呼ばれる遺伝的オペレータを各世代ごとにその個体の持つ染色体や遺伝子にそれらの操作を確率的に行う。GAは遺伝子をもつ仮想的な生物の集団を計算機内に設定し，予め定めた環境(条件)に適応している個体が子孫を残すように世代交代シミュレーションを実行することによって，遺伝子および生物集団を進化（適応）させていく。この生存競争に基づく確率的探査原理によって組合せ問題のような計算量の多い問題の近似解または最適解を比較的容易に短時間で得ることができるとされている。GAは実際のプログラミングの詳細を規定しない，緩やかな枠組みのため，各種の規則やパラメータの設定方法など，不確定要素が多い方法論である，と指摘されることもあるが，むしろ緩やかな枠組みであるため応用範囲が広いともいえる。

3. JSP

スケジューリングとは，ある目的を達成するため共通に使われる資源の時間配分を決定することである。JSP とは，複数の異なる仕事を処理するために，共通資源である機械群の時間的な割り当てを決定する問題ということができる。本研究で扱う JSP は次のように記述できる。n個の仕事（ジョブ）をm台の機械（マシーン）上で処理することを考える。各仕事を処理する機械の順序を技術的順序と呼び，技術的順序は仕事ごとに予め与えられている。各機械上での各仕事の処理は作業と呼ばれる。各作業は与えられた処理時間をかけて各機械上で中断なく処理される。ここで各機械はすべて異なり，同時に二つ以上の仕事を処理できず，逆に各仕事は一度に二つ以上の機械上では処理されないとする。すべての仕事を完成させるまでの時間を総作業時間（makespan）とよび，Lで表す。このとき，n×m（総作業時間最小）一般 JSP とは，各仕事の技術的順序と，各作業の処理時間が与えられたもとで，Lを最小にするような各機械上での仕事の処理順序をすべて決定することである[3,4]。

例えば図1の場合，ジョブ1（J1）はマシーン3（M3）からマシーン1（M1）…マシーン5（M5）の順にスケジュールしていくことを意味する。図2の場合，J1は最初にスケジュールされると処理時間が1かかり，2番目にスケジュールされると3，3番目にスケジュールされると6，…，6番目にスケジュールされると処理時間が6かかることを意味する。

すなわち，JSP とは，図1，図2のように予め与えられた制約条件を満たしながら各機械上での仕事の処理順序を決定し，すべての仕事をスケジュールさせるまでの時間であるLを最小化していくものである。

JSP の解空間は，探索空間の大きさの観点から図3のような包含関係で表すことができる。実行可能スケジュールとは技術的順序を満たすスケジュールであり，セミ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することなしにどの作業の開始をも早めることができないスケジュールである。また，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の変更を許したとき，どの作業も，他の作業の開始時刻を遅らせる

J1:	M3	M1	M2	M4	M6	M5
J2:	M2	M3	M5	M6	M1	M4
J3:	M3	M4	M6	M1	M2	M5
J4:	M2	M1	M3	M4	M5	M6
J5:	M3	M2	M5	M6	M1	M4
J6:	M3	M6	M2	M1	M5	M3

図1 技術的順序

J1:	1	3	6	7	3	6
J2:	8	5	10	10	10	4
J3:	5	4	8	9	1	7
J4:	5	5	5	3	8	9
J5:	9	3	5	4	3	1
J6:	3	3	9	10	4	1

図2 処理時間

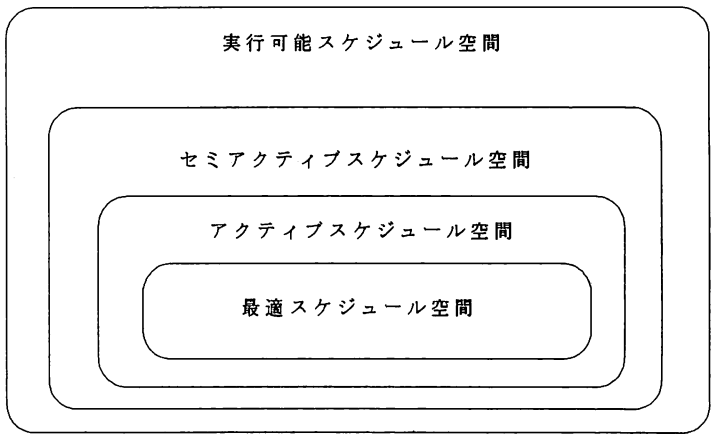


図3 JSP の解空間の構造

ことなしに，その処理開始を早めることができないスケジュールである[5]。

すべてのアクティブスケジュールを網羅的に生成する方法として，Giffler と Thompson によって提案された GT 法[2]がある。GT 法のアルゴリズムを以下に示す。

GT法

(1) 未スケジュールな作業のうち最早完了時間が最小のものを O^*とし，O^*を処理する機械を M^*とする。

(2) M^*上で処理され，かつ O^*と処理が重なる未スケジュールの作業からなるコンフリクト集合Cを作る。

(3) コンフリクト集合Cから作業を1つ選択し，スケジュールする。

(4) 未スケジュールの作業がなくなるまで，(1)～(3)を繰り返す。

上記のステップ(3)において，コンフリクト集合Cからの作業の選択の仕方には任意性が存在する。ここで，系統的に作業を選択することにより網羅的にアクティブスケジュールを生成でき，また，ランダムに作業を選択することにより任意のアクティブスケジュールを生成することができる。

4．GA による JSP の実装

本実験で実装される GA では，順序型交叉と位置型交叉の2つの交叉法を用いて以下のようなコード化，適応度関数，遺伝的オペレータを採用した。

4.1 処理手順

GA による処理手順は以下のようになる。

1．初期集団の生成

(a) コード化

2．終了条件が満たされるまでのループ

(a) 適応度の評価

(b) 選択

(c) 交叉

(d) 突然変異

4.2 コード化

JSPを解くにあたって，この問題の個体を文字列（染色体）にコード化しなければならない。コード化はその物体の数のビット長を要求する。JSPにGAを適用することにおける染色体表現は，遺伝的オペレータに影響を与える重要な段階である。本研究では，各マシーンごとの仕事列を表すPreference-list-Based表現[1)]を使用した。例えば図4の場合，M1においてはJ1→J4→J3→J6→J2→J5の順にスケジューリングしていくことを意味する。M2～M6についても同様である。

4.3 GT法による強制

前節4.2の図4のように生成される仕事列が必ずしもアクティブであるとは限らないのでGT法を用いる。つまり，仕事列がアクティブであれば，GT法は適用されず，仕事の入れ換えは行われない。アクティブでなければ，GT法が適用され違反箇所において仕事列がアクティブになるように仕事の入れ換えを行う。すなわち，アクティブスケジュールであるか否かの判定およびアクティブスケジュールへの強制の方法としてGT法を利用することを意味する。

4.4 適応度関数

適応度関数は，個々の個体の競争力を表し，適応度関数を用いて宣言的に問題を記述できる。本研究ではすべての仕事を完成させるまでの時間である総作業時間（makespan）により算出する。

（適応度関数）

$$\text{makenkan} = \min \max_{1 \le k \le m} \left\{ \max_{1 \le i \le n} \{c_{ik}\} \right\}$$

M1: J1 J4 J3 J6 J2 J5

M2: J2 J4 J6 J1 J5 J3

M3: J3 J1 J2 J4 J5 J6

M4: J3 J6 J4 J1 J2 J5

M5: J2 J5 J3 J4 J6 J1

M6: J3 J6 J2 J5 J1 J4

図4 Preference-list-Based 表現

4．5 遺伝的オペレータ

4．5．1 選択（selection）

集団における適応度に従って，次世代に生存する個体群を決定する。集団の中から個体を取捨選択するだけで，新しい個体を生成するものでない。新しい個体の生成は交叉と突然変異によって行われる。本研究では，エリート保存戦略を使用した。エリート保存戦略とは集団の中で適応度が最も良い値のものを次世代に残す方法である。

4．5．2 交叉（crossover）

交叉は，二つの親の染色体を組み替えて子の染色体を作る操作であり，二つの個体の間で文字列の部分的な交換を確率的に行う操作である。本研究では，位置の集合をランダムに選択し，一方の親の選択された位置におけるタスクの順序をもう一方の親に対応するタスクに課する順序型交叉および位置の集合をランダムに選択し，一方の親の選択されたタスクの位置をもう一方の親の対応するタスクに課する位置型交叉を使用した。

（順序型交叉）

- ランダムに位置を選択し，親1（または親2）の選択されたジョブ番号の順番に親2（親1）の順番を対応させて，それが子2（子1）になる交叉。

親1：	1	2	3	4	5	6	7	8	9
親2：	5	9	2	4	6	1	7	3	8
選択位置		*	*		*		*		
					↓				
子1：	1	9	3	4	5	2	6	8	7
子2：	2	9	3	4	6	1	5	7	8

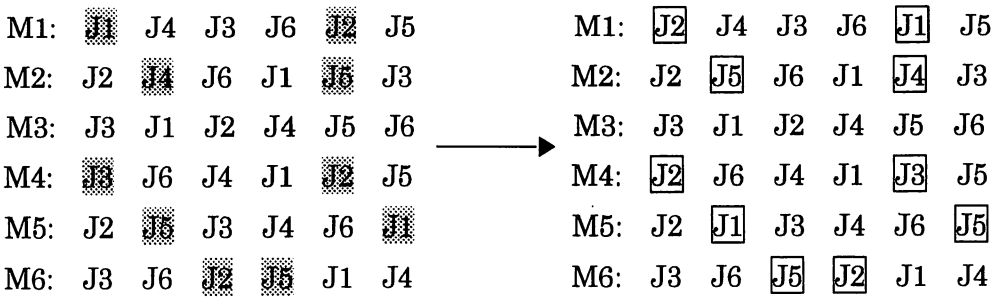
（位置型交叉）

- ランダムに位置を選択し，親1（親2）の選択された位置と親2（親1）の選択された位置を入れ替えて，親1（親2）で選択されなかったジョブ番号の順序を親2（親1）の順序に並び替え対応させ，それが子2（子1）になる交叉。

親1：	1	2	3	4	5	6	7	8	9
親2：	5	9	2	4	6	1	7	3	8
選択位置		*	*		*		*		
					↓				
子1：	1	9	2	3	6	4	7	5	8
子2：	9	2	3	4	5	6	7	1	8

4．5．3 突然変異（mutation）

突然変異は，遺伝子を一定の確率で変化させる操作であり，個体中の特定ビットを別のビットに確率的に変更する操作である。本研究では，ある機械上で割り当てられている任意のジョブを二つ選び，それらを入れ替える突然変異を使用した。なお，突然変異により生成される個体は必ずしもアクティブスケジュールであるとは限らないので，GT法による強制が必要である。



5．実験結果

本研究では，10ジョブ×10マシーンのベンチマーク問題に順序型交叉，位置型交叉を用いて比較検討を行うことを目的とした。個体数を50から500まで50ずつ変化させ，打ち切り世代数は個体数に応じて決め（500～10000世代），交叉確率は1.0，突然変異確率は0.01，試行回数は5回である。また10×10問題の最適解は930である。使用した計算機はPC-9821 V200，使用言語はC言語である。

表1，2は順序型交叉，位置型交叉において，個体数が50から500まで50ずつ変化させたものに対するmakespanと個体数に応じて決めた世代数までの計算時間とmakespanの収束率および評価回数と個体数によって決まる世代数を示したものである。表1の(a)～(c)は順序型交叉の各評価回数ごとの個体数とmakespanの関係を表し，表2の(a)～(c)は位置型交叉の各評価回数ごとの個体数とmakespanの関係を表している。表1，2の(a)，(b)，(c)の内容は，

(a) 評価回数が50000回のときの個体数に対するmakespanの関係

(b) 評価回数が250000回のときの個体数に対するmakespanの関係

(c) 評価回数が500000回のときの個体数に対するmakespanの関係

を示したものである。

なお，

評価回数＝個体数×世代数

収束率＝最適解/makespan

で求めるものとした。

6．考　　察

表1，2の(a)～(c)より個体数が増加すれば，makespanは必ずしも最適解に近づいているわけではないが，makespanは最適解に近い解を得る傾向にあると考えられる。評価回数が増えると，評価回数が同じであるにもかかわらず各個体数の計算時間に大きな差が生じる。これは，個体数が増えるとその分だけ計算に時間を費やす傾向があると思われる。しかし，makespanは評価回数が多くても最適解に最も近い値を得るわけではない。それは個体数によって異なる。表1の(a)では個体数が450，表1の(b)では個体数が500，表1の(c)は個体数が450のときに最も良い解を得ている。次に，表2の(a)においては，個体数が500，表2の(b)は個体数が350，表2の(c)も個体数が450のとき最も良い解を得る。よって，順序型交叉は個体数が450～500，位置型交叉は個体数350～500の範囲で良質な解が得られるものと考えられる。

図5の(a)～(c)は本研究で得られた結果で，順序型交叉および位置型交叉の世代数に対するmakespanの関係を各評価回数ごとの収束率が最も良い値を算出するときの個体数で表し，図5の(a)～(c)の内容は，

(a) 評価回数が50000回で順序型交叉の個体数が450，位置型交叉の個体数が500のときのmakespanと世代数の関係

(b) 評価回数が250000回で順序型交叉の個体数が500，位置型交叉の個体数が350のときのmakespanと世代数の関係

(c) 評価回数が500000回で順序型交叉の個体数が450，位置型交叉の個体数が450のときのmakespanと世代数の関係

を示したものである。

表1，2の(a)～(c)における各個体数のmakespanを比較すると，最も良い解を得ているのは表1の(a)の個体数が450のときである。よって，評価回数についていえば，回数を増やしても必ず良いmakespanの値を得るわけではないということがわかった。

7．結　　論

本研究はJSPにGAを適用し，順序型交叉・位置型交叉について各評価回数ごとに個体数の変化に対する適応度，計算時間，収束率，打ち切り世代数のデータを示した。その結果より，順序型交叉は表1の(a)において個体数が450のときmakespanが最も良い値を得た。次に，位置型交叉は表2の(b)における個体数が350と表2の(c)における450のときともに最も良いmakespanの値が算出されている。そして順序型交叉と位置型交叉の最も良い収束率を比較すると順序型交叉のほうが少しだけ良いが，収束率に差のないことや個体数によって位置型交叉のほうが良い収束率を得ることもある。したがって，本研究で用いた2つの交叉法（順序型交叉・位置型交叉）は汎用的な交叉法であり，JSPにおいてはどちらが有効な交叉法であるのかが言えない。よって，JSPに用いる交叉法は汎用的な交叉

法ではなく，JSPを考慮に入れた交叉法のほうがより良い解が得られるのではないかと考えられる。

なお本研究では，JSPにおいて順序型交叉・位置型交叉についてのみ実験を行ったが，他の交叉法の有効性についても今後検討する予定である。

参考文献

1）M. Gen and R. Cheng：“Genetic algorithms and engineering design”, A Wiley-Interscience Pub., pp. 190-233, 1997.

2）B. Giffler and G. L. Thompson：“Algorithms for solving production scheduling problems”, Operations Research, vol. **8**, no. **4**, pp. 487-503, 1960.

3）山田武士，中野良平：“遺伝アルゴリズムとスケジューリング問題”，システム/制御/情報，vol. **37**, no. **8**，pp. 484-489, 1993.

4）山田武士，中野良平：“遺伝的局所探索法によるジョブショップスケジューリング問題の解法”，情報処理学会論文誌，vol. **38**, no. **6**，pp 1126-1138, 1997.

5）小野　功，佐藤　浩，小林重信：“サブシーケンス交換交叉とGT法に基づくジョブショップスケジューリングの進化的解法”，電気学会論文誌，vol. **117**-C, no. **7**，pp. 888-895, 1997.

表1(a) 順序型交叉の個体数と makespan の関係（評価回数：50000回）

個体数	makespan	計算時間（秒）	収束率（%）	世代数
50	1074.8	20.026	86.52	1000
100	1043.2	22.75	89.14	500
150	1049.6	24.03	88.6	333
200	1063	24.338	87.48	250
250	1036	26.49	89.76	200
300	1040.4	29.562	89.38	167
350	1037.2	31.966	89.66	143
400	1034.2	33.618	89.92	125
450	1026	37.036	90.64	111
500	1057.4	39.93	87.95	100

表1(b) 順序型交叉の個体数と makespan の関係（評価回数：250000回）

個体数	makespan	計算時間（秒）	収束率（%）	世代数
50	1076.4	111.11	86.39	5000
100	1057.8	116.948	87.91	2500
150	1063.2	111.028	87.47	1667
200	1033.2	122.7	90.01	1250
250	1044.6	130.868	89.02	1000
300	1030.4	135.86	90.25	833
350	1040.2	148.38	89.4	714
400	1037.4	162.328	89.64	625
450	1034.8	169.482	89.87	556
500	1027.6	176.014	90.5	500

表1(c) 順序型交叉の個体数と makespan の関係（評価回数：500000回）

個体数	makespan	計算時間（秒）	収束率（%）	世代数
50	1068.4	201.082	87.04	10000
100	1050.4	303.716	88.53	5000
150	1058.4	243.144	87.86	3333
200	1045.2	249.956	88.97	2500
250	1054	263.064	88.23	2000
300	1034	280.386	89.94	1667
350	1042.8	311.324	89.18	1428
400	1038.8	338.586	89.52	1250
450	1028	349.296	90.46	1111
500	1038.6	363.228	89.54	1000

表2(a) 位置型交叉の個体数と makespan の関係（評価回数：50000回）

個体数	makespan	計算時間（秒）	収束率（%）	世代数
50	1061.2	19.934	87.63	1000
100	1063.4	24.176	87.45	500
150	1050.2	25.332	88.55	333
200	1042	25.998	89.25	250
250	1035.4	27.862	89.82	200
300	1054.2	30.564	88.21	167
350	1050.6	32.502	88.52	143
400	1048.4	35.446	88.7	125
450	1031.8	38.456	90.13	111
500	1031.2	40.734	90.18	100

表2(b) 位置型交叉の個体数と makespan の関係（評価回数：250000回）

個体数	makespan	計算時間（秒）	収束率（%）	世代数
50	1064.8	104.23	87.34	5000
100	1051.4	114.692	88.45	2500
150	1037.2	120.95	89.66	1667
200	1045.2	124.748	88.97	1250
250	1034.6	134.184	89.88	1000
300	1033.2	168.498	90.01	833
350	1028.8	165.912	90.39	714
400	1034.2	179.25	89.92	625
450	1037.4	189.226	89.64	556
500	1029.4	191.578	90.34	500

表2(c) 位置型交叉の個体数と makespanの関係（評価回数：500000回）

個体数	makespan	計算時間（秒）	収束率（%）	世代数
50	1059.4	215.17	87.78	10000
100	1069.8	218.006	86.93	5000
150	1049.6	232.74	88.6	3333
200	1047.4	275.776	88.79	2500
250	1047.4	245.478	88.79	2000
300	1039.6	278.19	89.45	1667
350	1037.6	294.224	89.62	1428
400	1035.8	325.508	89.78	1250
450	1028.8	341.018	90.39	1111
500	1044.2	377.92	89.06	1000

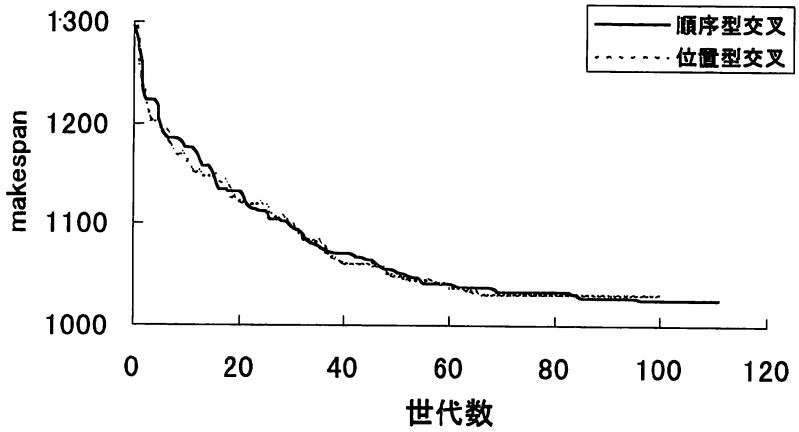


図5(a) 2つの交叉の makespan と世代数の関係（評価回数：50000回）

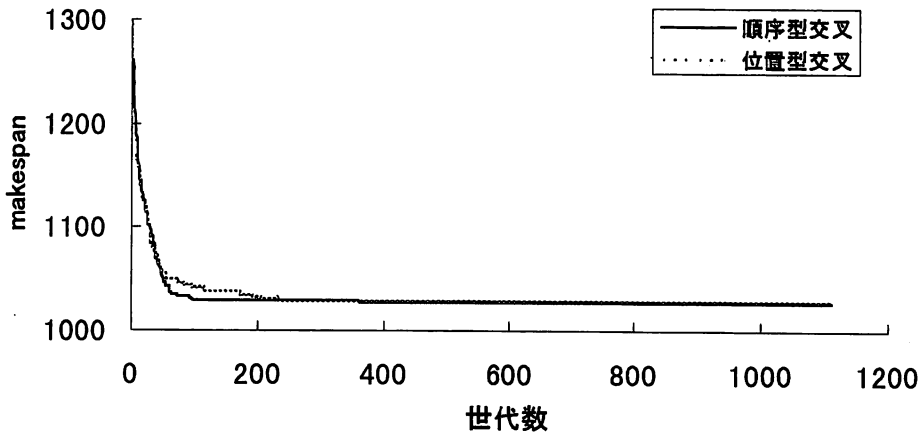


図5(b) 2つの交叉の makespan と世代数の関係（評価回数：250000回）

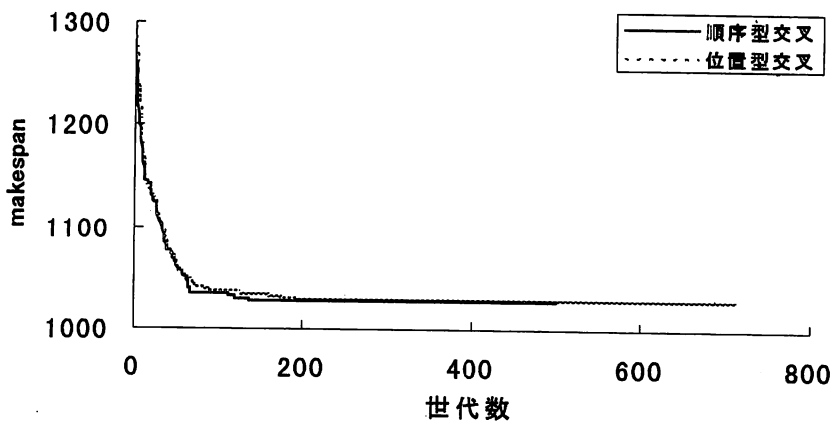


図5(c) 2つの交叉の makespan と世代数の関係（評価回数：500000回）

Some Comments on Efficient Crossover for Schedule Problem

Toshiharu WAKAMIYA, Kengo KATAYAMA and Hiroyuki NARIHISA*
Graduate School of Engineering
**Department of Information & Computer Engineering*
Okayama University of Science,
Ridai-cho 1-1, Okayama 700-0005, Japan
(Received October 5, 1998)

Recently, heuristic procedures have been recognized as competent procdures for combinatorial optimization problems.

As is already known, almost combinatorial optimization problems are reduced to NK-hard problem; that is to say, it is impossible to get an exact optimal solution within polynomial time. However, there may occur frequently the case that even though the near optimal solution (not an exact optimal solution) is necessary in practical engineering problems. In these circumstances, heuristic algorithm is often used.

In this paper, we study about the efficient genetic abgorithm for solving the scheduling problems. Genetic algorithm is also a kind of heuristic procedures, and many success examples are reported. From the above mentioned reasons, we investigate the efficient crossover as a genetic algorithm's operation for a job-shop scheduling problem.