Heap法に基づくブロック組立計画自動化に関する研究

岩下, 寛弥

https://doi.org/10.15017/1785386

Heap 法に基づく
ブロック組立計画自動化に関する研究

九州大学大学院　工学府海洋システム工学専攻

岩下　寛弥

2016 年 7 月

目次

第 1 章 序論

1.1 本研究の目的

日本の造船業を取り巻く環境は益々厳しさを増している。他国の造船業の台頭や為替変動といった受注環境の変化に適応するためにも、更なる生産性向上が必要とされる。そのために考慮しなければならない問題として、物流管理と精度管理が挙げられる。船舶建造における造船所内物流管理は、1 隻数万点におよぶ部材を供給する切断工場から始まり、1 隻数十個のブロックが組立ラインおよび総組場を経て、ドック内で 1 隻の船舶となるまで一貫して管理していく必要があり非常に困難を極める。

物流管理における理想形はジャストインタイム・スケジューリングである。トヨタ生産方式に代表されるジャストインタイムの理念は、ある部材・部品が必要とされるタイミングに、その部材・部品を完成させることで、無駄や無理をなくし生産効率を向上させるものである。造船分野に対して、ジャストインタイムを適用する場合に、最も重要となるのがブロック組立日程工程である。前船の進水と次船の受注の関係から、ドック内作業期間が決まる。それに応じて、ブロック搭載の順序と開始日が確定する。そのため、この搭載日に対してジャストインにブロックを製作する計画を立てることが、生産効率向上の要となる。

ブロック組立工程計画立案時には、4 つの要件を考慮する必要がある。

第 1 要件は、設備制約を満足させることである。造船に必要なブロックが巨大でありまたブロックの制作に必要な部材も膨大となる。そのため組立定盤の数やストックヤードの蔵置区画数を適切に管理した計画を立案しなければ、すぐに溢れを引き起こす。

第 2 要件は、ライン間の同期制約である。ブロック組立工程は複数の工程ラインが平行して存在し、1 つのラインにブロックや作業人員が集中しないように計画を立てる必要がある。この際に、工程ライン間で作業開始や完了のタイミングを調整しなければ、次工程への作業に滑らかに移行できない。

第 3 要件は、ブロックの製作順序である。ブロックを製作する順番が、無駄な待ち時間であるアイドル期間や全体の作業期間であるリードタイムの短縮に影響を与える可能性がある。これは、多くの人員を必要とするブロックが同時期に重なると、造船所の抱える作業員数を超えてしまい、追加人員の費用が発生することからも重要となる。

第 4 要件は、作業定盤の選択である。組立工程やストックヤードは複数の定盤や区画を擁している。このとき、ブロックを組み立てる定盤を適切に設定しなければ、リードタイムが伸長した計画となってしまう。

このように、工程計画立案作業は複数の要件が絡み合っており、建造期間や建造効率、

人件費といったコストに直結する重要な作業である。しかし、現状では熟練の作業者が過去の経験から手作業で、上記の要件を満たすように計画を作成している。これには煩雑な作業と数ヶ月の作業期間を必要とする。そのため、扱いやすいモデルでブロック組立作業を表しつつ単純な処理によって計画作業が行える、体系だった計画手法が必要とされている。

そこで本研究では、ジャストインタイム・スケジューリングを取り込んだ新しい計画手法として、Heap モデルに基づく計画手法を提案した[1],[2]。この計画手法を実造船所のブロック組立工程データに適用することで、その実用性を確認することを目的とする。

以下では、1.2 節で研究対象である造船所 A および造船所 B の組立工程と工程計画の現状について述べ、1.3 節で工程計画の定式化を行い工程計画の目的関数の定義や、制約条件の確認を行う。

1.2 造船所における工程計画の現状と問題点

ここでは本研究で扱う、造船所 A および造船所 B の工程計画について、現状の調査および問題点の抽出、定式化を行う。

まず各造船所のブロック組立工程について把握する。そのために造船所内のブロック製作の流れを調べ、使用する設備の詳細や工程計画で取り扱う範囲を求めた。次にブロック組立工程計画の現状について調査し、各造船所の抱える問題とその共通点を抽出した。最後にブロック組立工程計画を定式化し、組立工程に対して汎用的に適用できる工程計画手法を提案するために解決する必要のある制約条件をまとめた。

1.2.1 造船所 A のブロック組立計画

研究対象のある造船所 A のブロック組立工程計画の現状について説明する。この造船所 A の外観図を Fig.1-1 に示す。

この造船所は、船舶の建造方式にツインタンデム建造方式を採用している。ツインタンデム建造方式とは、建造ドック内に 2 隻の船舶を並列して 1 組とする。さらにこの組を縦に 2 組並べ、2 組 4 隻をひとつのサイクルとして建造する方式をとる。当該造船所は主に同型船を扱っており、年間の建造隻数から建造ピッチ日数を決定し、それに応じて各サイクルの搭載日程が定まる。ブロック搭載は短期間に行われるため、その期間に必要なブロックは予め総組を終えておく必要がある。

この造船所では、大組ブロックと中組ブロックの組立工程ラインはコンベア化されている。このコンベアラインの写真を Fig.1-2 に示す。ラインごとに定められたピッチ時

間が経つと定盤を移動させる。ブロックの組み立てが完了すると定盤はコンベアの終端にいるため、そのまま出棟することになる。コンベアラインを使用しているために、この出棟がコンスタントに行われることが特徴のひとつとなっている。

Fig.1-1 造船所 A の外観図

Fig.1-2 コンベアライン

コンベアから出棟したブロックはストックヤードに運ばれる。このストックヤードの写真を Fig.1-3 に示す。ストックヤードは大組ブロック用と中組ブロック用の区画がそれぞれ割り当てられている。この区画は 2 隻分のブロックが蔵置できるだけの数が用意されている。そのため年間の建造隻数を増加させ建造ピッチを短縮すると、コンベアからストックヤードへの搬入数と、ストックヤードから次工程、特に総組場への搬出数が適切に管理されなければ、ストック区画不足が懸念される事態となる。

このように各設備の許容量を超えないように設備制約を守った計画を行うことに加えて、この造船所の特徴である先行中組ブロックの問題が計画作成を困難にする。

先行中組ブロックは、一旦中組ブロック組立ラインを通過しストックヤードにて先行艤装を行ったのち、ふたたび中組ブロック組立ラインへ投入されるブロックである。この先行中組ブロックは、同じラインに再投入されるとは限らず、別の中組ブロック組立ラインに投入されることもある。中組ブロック組立工程の工程計画作成の際は、先行中組・通常中組の組立開始日と完了日を調整する必要があり非常に煩雑な作業となる。これは複数ライン間の同期をとる制約と考えられるため、本研究ではこの制約を同期制約と呼ぶ。

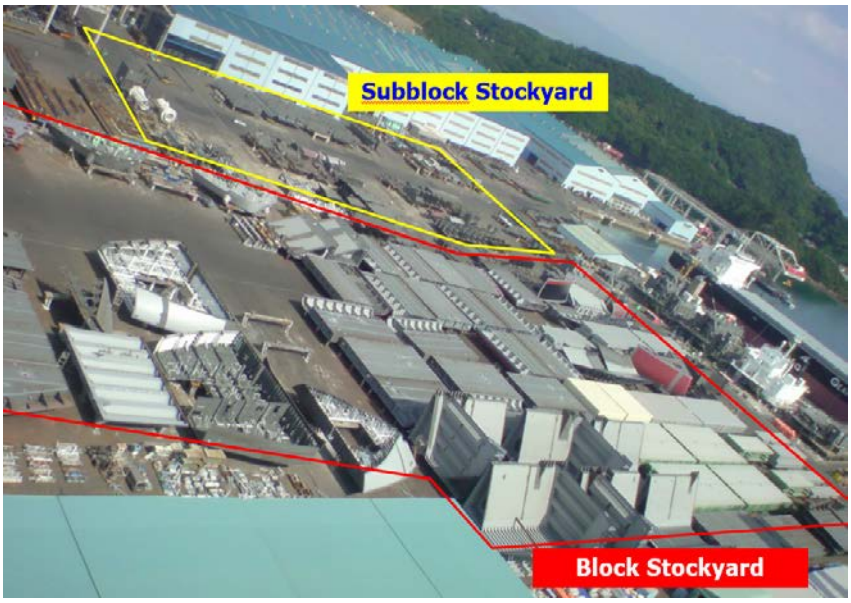

Fig.1-3 ストックヤード

このような状況のなか、当該造船所では表計算ソフト Excel を利用した手作業による工程計画を行っている。この工程計画用シートを Fig.1-4 に示す。これは組立コンベアラインの工程計画を行っているものである。縦軸にコンベア区画、横軸に日数を取る。各矢印が作業期間を表しており、矢印上にブロック名が記載されている。各日に対して縦方向に矢印を数え上げるとその日にコンベア定盤を占有しているブロックの数がわかる。これがブロック個数の欄に記されたもので、定盤数 9 区画を超える日は赤で警告される。この図では、矢印の表すコンベア作業期間が 4.85 日を 5 日としているため、数値上の重なりが生じており 10 区画となっている。そのためこの程度の定盤数オーバーは許容される。このようにして設備の制約を満たしつつも、ライン間の同期制約を満足しなければならず、またブロック数も膨大となるため、計画作成作業は非常に煩雑な作業内容である。

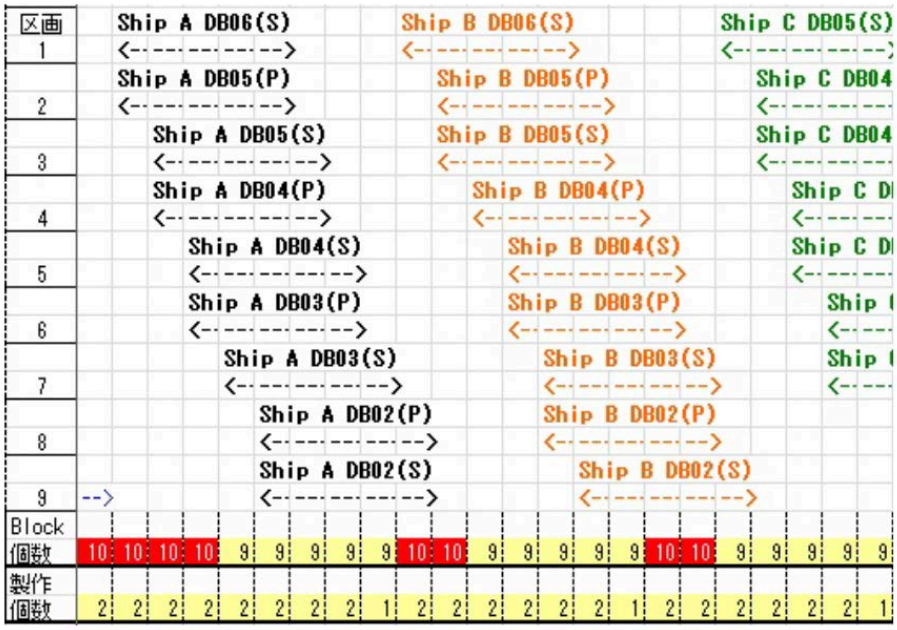

Fig.1-4 コンベアラインの工程計画に用いる表計算シート

1.2.2 造船所 B のブロック組立計画

研究対象のある造船所 B のブロック組立工程計画の現状について説明する。この造船所 B の外観図を Fig.1-5 に示す。

Fig.1-5 造船所 B の外観図

この造船所は、一般的な造船所に比べて敷地面積が小さいことが特徴である。鋼材の切り出しから建屋内作業場、屋外作業場と続く一連の組立工程において、この造船所には各工程の間でバッファとして働くストックヤードが存在しない。屋外作業後から塗装工場の間および塗装完了後から搭載までの間、海上に浮かべた台船(バージ)にブロックを蔵置する。そのため、適切なタイミングでブロックの搬入出を行う計画が必要となる。

この造船所の組立工程計画作成の流れについて説明する。船舶の設計が終わると、CAD データから Fig.1-6 に示すブロック形状表を作成する。縦軸にブロック名、横軸に形状がまとめられており、縦・横・高さの値がまとめられている。このブロック形状表をもとに Fig.1-7 に示す組立日程表を作成する。縦軸にブロック名、横軸に日数をとり、組立開始から搭載日までのすべての作業をまとめる。このとき各作業にかかる日数は、ブロックの形状やブロック重量と過去の経験を参考にしたものである。

組立日程表が完成すると、Fig.1-8 に示すブロック日程表を作成する。このブロック日程表は縦軸にブロック名、横軸に作業内容と作業区画、開始日、作業日数を記入する。この日程表は組み立てるブロックを中心に作成されているが、さらに組み立てる場所に主眼をおいたブロック定盤日程表を作成する必要がある。これを Fig.1-9 に示す。これは縦軸に各作業定盤を表示し、横軸に日数をとる。各定盤でどのブロックのどんな作業が行われるか、この日程表を見て確認する。

以上の日程表が完成すると、この造船所の保有するツールを用いて、Fig.1-10 に示すブロック定盤配置図を得ることができる。これは CAD データ上に描かれた造船所の設備地図の上に、組立工程の作業定盤をブロックがどう通過するか視覚化したものである。この配置図を毎日工場内に貼りだすことで、作業者全員がブロックの現在地と次工程の位置を把握することができる。

本来このブロック定盤配置図を作成するまでが、工程計画の目標である。しかし現在、この造船所でこの工程計画に携わる作業者は 1~2 名で、ブロック定盤日程表を作成するまでに 2 ヶ月近い期間を必要とする。これは最後のブロック定盤配置図を作成する以外の作業がすべて作業者による手作業あることや、その作業を行うために造船所のこれまでの実績などを把握している熟練者のみに頼っているためである。また場合によっては、ブロックを組み立てる作業定盤を工程計画の時点では確定できず、そのブロックの組み立て直前に現場の作業者に指示する状況が発生している。

船番	1741		船種	5,500DWT積み　白油タンカー							
			取込		作成		登録		進水日	12月15日	
									搭載日	8月7日	

ブロック名	フラグ	形状									
		タイプ	A	B	C	D	E	F	G	H	I
1S1	0	C	8,000	5,000	1,000	1,200					
2S1S	0	B	7,000	6,500	800						

Fig.1-6 ブロック形状表

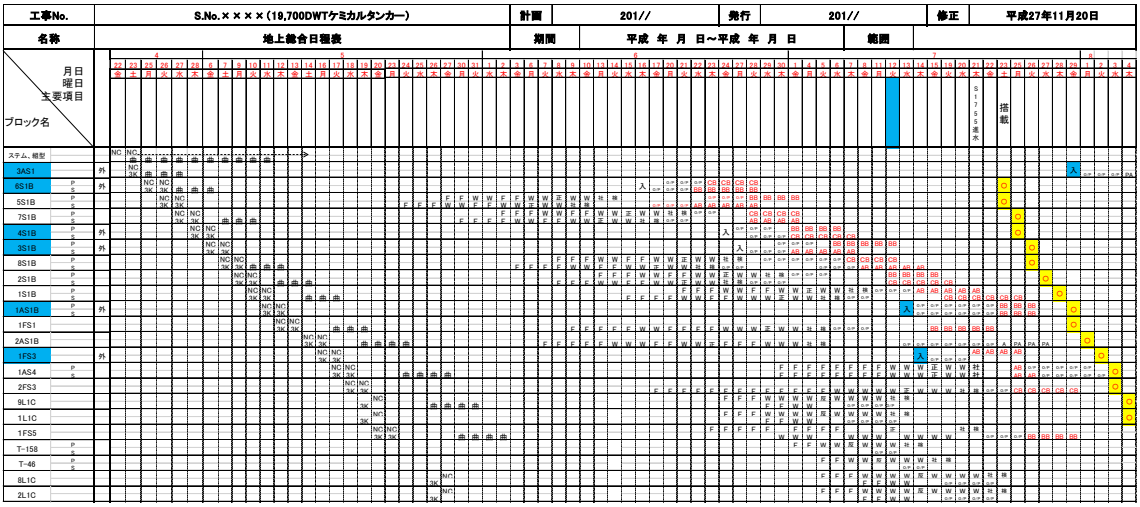

Fig.1-7 組立日程表

船番	×××	船種	19,700DWTケミカルタンカー			取込				②-1	インポート			②-2	エクスポート		

ブロック名	搭載日	日程①				日程②				日程③				日程④			
		ステージ	職種	開始日	日数	ステージ	職種	開始日	日数	ステージ	職種	開始日	日数	ステージ	職種	開始日	日数
4S1	6月20日	K1-2	F	4月23日	6	K3-4	W	5月7日	11	S3-1	O/P	5月22日	3	B1	B	5月28日	4
3S1	6月20日	K1-1	F	4月24日	5	K3-3	W	5月7日	13	S3-2	O/P	5月27日	3	B2	B	5月30日	4
5S1	6月20日	K1-2	F	5月7日	5	K3-2	W	5月14日	11	S3-1	O/P	5月29日	3	B1	B	6月3日	4
2S1	6月21日	K1-2	F	5月14日	4	K1-4	W	5月20日	12	S3-1	O/P	6月5日	3	B1	B	6月10日	4
1S1	6月21日	K1-1	F	5月8日	4	K3-1	W	5月14日	11	S3-3	O/P	5月29日	3	B3	B	6月3日	4

Fig.1-8 ブロック日程表

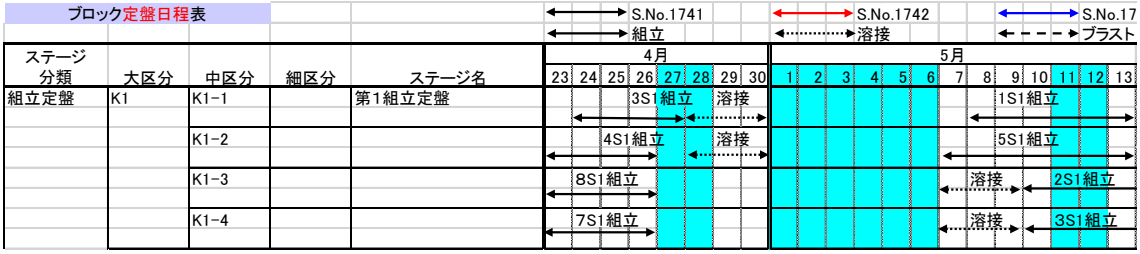

Fig.1-9 ブロック定盤日程表

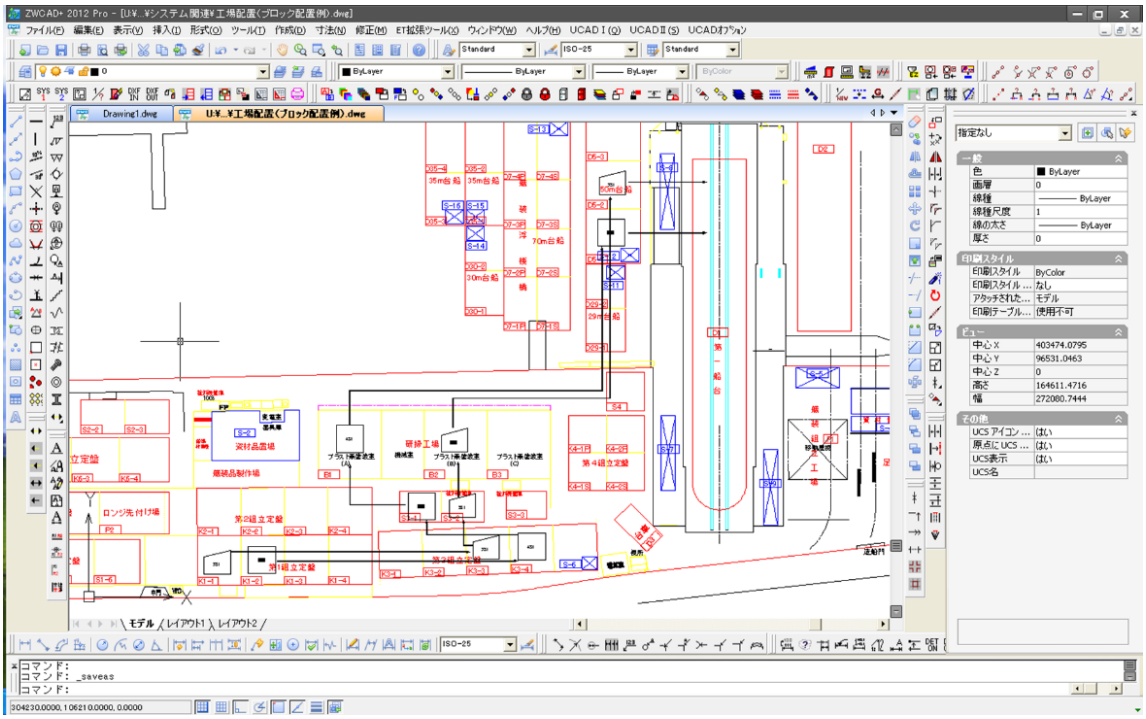

Fig.1-10 ブロック定盤配置図

1.2.3 工程計画の抱える問題点

両造船所の抱える問題点の１つ目が、生産工程の制約問題である。造船所のブロック組立工程は固定定盤もしくはコンベアラインが用いられる。コンベアラインには移動定盤式のものとチェーンコンベア式のものがあるが、ここでは移動定盤式を想定する。ブロック組立工程計画を立てるには、これら組立工程設備の定盤数を超えないように日々のブロック数を調整する必要がある。

組立工程を通過したブロックは、次の工程に直接搬入される場合と、蔵置区画に一時蔵置される場合がある。この保管区画として一般的にストックヤードが用いられるが、海上に浮かべた台船をストックヤードの代用とする造船所もある。船舶を構成するブロックは、自動車などの部品と異なり、非常に大きく重量物であるためラックにまとめることができず、蔵置するだけでも場所を広く占有する。組立工程から搬出されたブロックと、次の工程へ搬入するブロックの数を調整しつつ蔵置区画での蔵置計画を立てなければ、蔵置区画のオーバーフローを引き起こすことになる。

このような設備の許容量内に収まることを、設備制約と呼ぶ。設備能力の面から実行可能な計画を立てるためにはこの設備制約を遵守する必要がある。

もうひとつの制約条件として同期制約問題がある。組立工程は複数の工程から成り立っており、ある工程で必要なブロックを並行する組立工程や、隣接する定盤で製作する

ことが常となっている。そのため、あるブロックや部材の完成期日と、それを必要とするブロック組立作業の開始日が逆転するような計画が出来てはならない。このように複数の組立工程間や複数の作業間の同期のとれた、製作順序に沿った計画を立てるために同期制約を遵守する必要がある。

本研究では、制約問題に関して以上の 2 つの制約を考慮する。

両造船所の抱える問題点の 2 つ目が、生産計画の煩雑さである。中小規模の造船所では、生産計画の作成におもに表計算ソフト Excel を用いている。表計算シート上に描写した矢印や文字によるガントチャートを操作して、制約条件を満たしているか確認しつつ手作業で計画を作成する。大規模な造船所では、自社製のスケジューリングツールを用いている造船所もあるが、ガントチャートの表示と作業工数の数え上げといった表計算ソフトで行っていた処理の発展版で、その操作は手作業であることが多い。

大手企業であれば、スケジューリングに関わる部署の人員を多く雇用することができ、大人数で作業を分担することができる。しかし中小規模の造船所では、工程計画に関わる人員が少なく 1 隻分の計画作成だけでも長い期間を必要とする。またスケジューリング作業が長年の経験と勘で行われているため、その計画の最適性について疑問が残る。

そのため論理的な計画手法を背景とした、自動スケジューリングツールによってこれらの問題の解決をはかる。

1.2.4 工程計画問題の定式化

造船所の組立工程は、ブロック組立工場とバッファとなる蔵置区画に大別される。蔵置区画では単純に蔵置されているだけでなく、先行艤装や先行塗装といった作業を行う。従って、各製品(ブロック)を機械 1(組立工場)と機械 2(蔵置区画)の順番で製作する、一種のフローショップ・スケジューリング問題と考えることができる。これは一般に次のような適当な評価関数を最小とする組合せ最適化問題

Minimize　　評価関数

Subject to　　制約条件 & 優先規則

として定式化される。フローショップ・スケジューリング問題では評価関数として様々なものが考えられるが、本研究ではプル型スケジューリングを実現するために、各ブロックのアイドル期間をできるだけ小さくすることを考える。このアイドル期間は、現工程の作業完了日から次工程の作業開始日の間にある特に作業の行われていない時間である。そのため次の評価関数を設定する。

評価関数　　：各ブロックのアイドル期間の総和

また制約条件は次のように表される。

制約条件 1　　：組立工場の定盤使用数が一定値を超えない

制約条件 2　　：蔵置区画の区画使用数が一定値を超えない

制約条件 3　　：ブロック組立作業の順序が前後しない

制約条件の 1 と 2 は、組立工程の設備制約について述べたものである。制約条件 3 は同期制約について述べたものである。

この評価関数は、製品の製作順序と製作場所に依存している。そのため製作順序および作業定盤の選択を考慮する必要がある。ある組立工程である期間に製作するブロックの数をN個とすると、製作順序は$N!$通り考えられる。組合せ最適化手法を使用する場合は、この製作順序のなかからアイドル期間を最小化するものを探索する必要がある。もし計算を簡易化したい場合、どの製作順序を優先するかの規則として、

優先規則 1　　：ある規則に基づき生成される製作順序

を利用することになる。

ある組立工場に作業定盤がM定盤あるとする。先程と同様にN個のブロックを組み立てる場合に、どの定盤でブロックを製作するか指定するとその組合せはM^N通りのパターンが存在する。こちらに関しても組合せ最適化手法で探索を行わない場合、計算を簡易化するために優先規則を利用することになる。

優先規則 2　　：ある規則に基づき生成される定盤選択

このような定式化のもとに、本研究では計画手法を提案し、中造船所 B のブロック組立工程計画の最適化を行っていく。

1.3 本研究のアプローチ

1.3.1 従来研究の限界

造船所の生産管理については，奥本による解説[3]，[4]，山崎による PERT に基づく研究[5] や、青山らによるペトリネットに基づく研究[6] などがある。しかしながら、以下に説明するように、PERT やペトリネットで、前節で定式化した問題を解くのは難しいと言わざるをえない。

一般に、スケジューリングとは事象を生起させる順序やタイミングを管理することを指し、システム理論上は離散事象システムとしてのモデリングと制御（管理）の問題となる。そのモデリングの代表的な枠組みとしては、時間付オートマトンによるものと時間付ペトリネットによるものがある。前者は、有限個の状態が事象によって遷移する状況をモデリングでき、特性解析に基づくスーパバイザリ制御などの制御方式が提案されているが、後者に比べてモデル化の能力が劣るとされている。一方、後者は、文献[6]で試みられたように、造船工場をドック、大組立工場、中組立工場、小組立工場、加工

工場に分けて、それぞれに変換・運搬・停滞などの作業をモデリングできる。ただ、ペトリネットによるモデルを用いて何らかの制御（管理）問題を解こうとすれば、それに基づくシミュレータに対するパラメータ最適化手法の援用が避けられないと思われる。オートマトンとペトリネットは、作業の並行動作ばかりでなく、デッドロックなどの競合動作までモデル化できる。しかしながら、一般に工場は構造化された既知環境であるから，競合動作は予め管理できるとしてよい。むしろジャストインタイム・スケジューリングの観点から，大量のブロック組立開始日を決めるための計算法を得るためのモデリングと制御（管理）の方法論が期待される。

PERT(Program Evaluation and Review Technique)[7]は、大規模な開発工程に対する計画･管理の手法として生み出された。PERT では個々の作業を矢線(アロー)で表し、それらの順序関係を盛り込んだ全工程のアローダイアグラム(矢線図)を作成する。各矢線には所要日数が付されており、これにより全体の工期を計算することができる。この特徴により複数の作業が同時進行するとき、作業間の順序関係や時間的な余裕の有無を明らかにした計画を作成でき、CP(Critical Path)を抽出することで工期を決定する重要な作業の系列を明らかにし、人員平準化を考慮しつつ工程計画を行うことができる。

PERT を用いて搭載計画を立てることは行われたが、この手法を組立工程へと適用しようとすると、設備制約を満足させるための作業が非常に煩雑になることが予想できる。

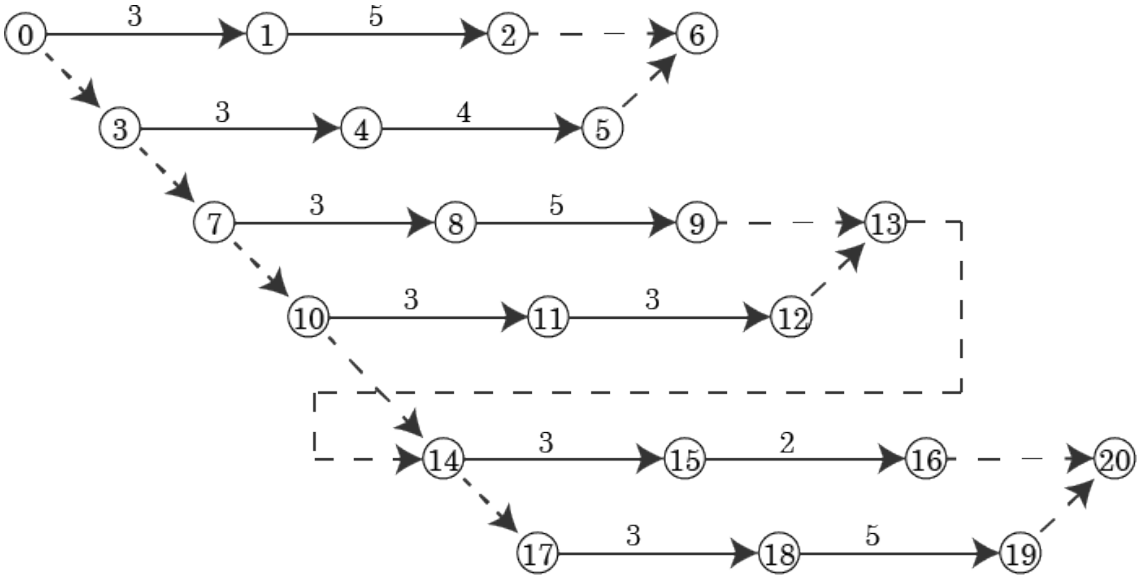

Fig.1-11 コンベア化された組立工程のアローダイアグラム

例えば、前述した造船所 A の組立工程を模した例を考える。コンベアラインとストックヤードからなる組立工程を仮定する。このコンベアラインでの作業は 3 日間行われ、コンベア定盤は 1 日に 1 定盤移動する。コンベアでの作業が完了するとストックヤードで艤装作業が行われる。この組立工程で組み立てる 4 つのブロックを考える。このブロ

ックのうち 1 つは 2 つのブロックを組み合わせて出来上がるブロックで、このベースブロックも同一ラインで作成するため、組立工程を通過するブロックは 6 ブロックとなる。これらに関する一連の作業をアローダイアグラムで表そうとすると、例えば Fig.1-11 のようなものが考えられる。このアローダイアグラムの矢線のうち実線が表す作業と所要日数を Table 1-001 に、ダミーを意味する破線が表す作業と所要日数を Table 1-002 に示す。

ブロックの組立は前述の通りコンベアラインとストックヤードを通過する。そのため、例えば Block 1 について矢線(0,1)でコンベア作業を矢線(1,2)でストック作業を表している。次のブロックである Block 2 はコンベア定盤が移動した翌日に搬入することができるので、1 日分のピッチをダミーアロー(0,3)で表現している。また、Block 3 は組立にベースとなるサブブロック Block 3-1 と Block 3-2 を必要とする。そのためこれらを先に組み立てた後に Block 3 の組立が始まるようにダミーアロー(13,14)で製作順序を固定している。

Table 1-001 アローの表す作業内容

結合点番号	作業内容	所要日数
(0,1)	Block 1 コンベア作業	3 日
(1,2)	Block 1 艤装作業	5 日
(3,4)	Block 2 コンベア作業	3 日
(4,5)	Block 2 艤装作業	4 日
(7,8)	Block 3-1 コンベア作業	3 日
(8,9)	Block 3-1 艤装作業	5 日
(10,11)	Block 3-2 コンベア作業	3 日
(11,12)	Block 3-2 艤装作業	3 日
(14,15)	Block 3 コンベア作業	3 日
(15,16)	Block 3 艤装作業	2 日
(17,18)	Block 4 コンベア作業	3 日
(18,19)	Block 4 艤装作業	5 日

Table 1-002 ダミーアローの表す作業内容

結合点番号	作業内容	所要日数
(0,3)	コンベア定盤移動	1 日
(2,6)	納期へダミー	0 日
(3,7)	コンベア定盤移動	1 日
(5,6)	納期へダミー	0 日
(7,10)	コンベア定盤移動	1 日
(9,13)	納期へダミー	0 日
(10,14)	コンベア定盤移動	1 日
(12,13)	納期へダミー	0 日
(13,14)	製作順序調整	0 日
(14,17)	コンベア定盤移動	1 日
(16,20)	納期へダミー	0 日
(19,20)	納期へダミー	0 日

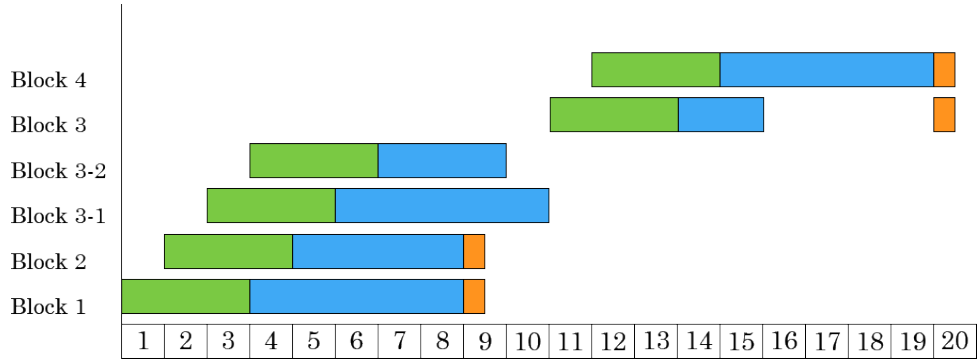

Fig.1-12 最早計画のガントチャート

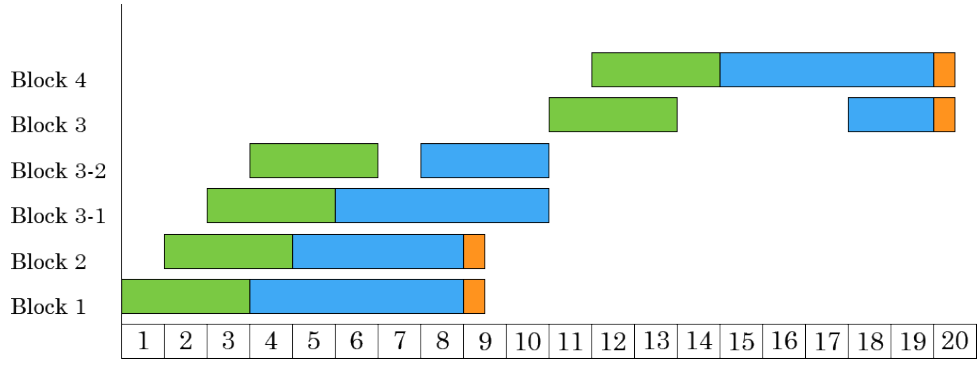

Fig.1-13 最遅計画のガントチャート

このアローダイアグラムをもとに最遅計画と最早計画を計算し、ガントチャートとしたものを Fig.1-12 と Fig.1-13 に示す。緑の横棒がコンベア作業を、青の横棒がストック作業を表しており、橙は納期となっている。コンベアの特徴である 1 日ごとの定盤の移動が表現できており、また Block 3 のサブブロックである Block 3-1 と Block 3-2 が、Block 3 の作業が始まる前に完成する計画を得ることができている。PERT ではここから人員平準化やストックヤードの許容量を守るような計画となるように、最遅と最早を比較して動かせる作業を調べ、調整を行っていく。ここの例ではブロックの数が 6 個であるため、作業の負担は軽い。しかし実際の組立工程では、工程計画に関わるブロックの数が 100 は超えるため、調整作業が非常に煩雑であることが予想できる。

更に、ブロックの製作順序を変更して計画を行いたい場合、Fig.1-11 に示すアローダイアグラムを組み直し、最早計画と最遅計画を立てる必要がある。このアローダイアグラムの組み直しにも非常に手間がかかることが予想され、効率的な組立工程計画の作成を行うことが難しい。

梶原ら[8],[9]は、造船工程を Max-Plus 代数を用いてモデル化を行った上で，ジャストインタイム・スケジューリング問題をある連立一次不等式の最大解として求めることを提案した。

PERT では作業をアローで表現したが、この Max-Plus 法では工程ラインの入出力モデルを作成し、そこにブロックを流すことで工程計画を行う。Fig.1-14 に Max-Plus 法でもちいる工程の 2 ポートモデルを示す。

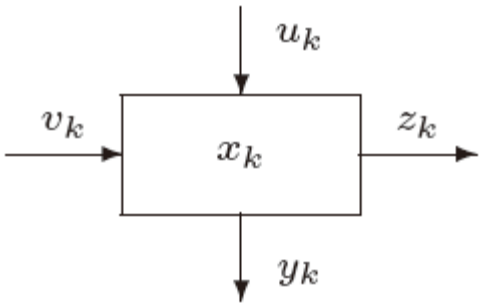

Fig.1-14 工程の 2 ポートモデル

２つの入力変数は部材の到着時刻 u_kと設備の占有開始時刻 v_k、２つの出力変数は作業の終了時刻 y_kと設備の開放時刻z_k、状態変数は作業の開始時刻x_kである。また d_kは作業時間を、c_kは占有時間を表す。この 2 ポートモデルを複数並べ、つなぎ合わせることでコンベアラインやストックヤードを表現する。モデルの入出力で作業開始や終了日だけでなく設備の占有も扱うために、設備制約を遵守した工程計画が可能となる。

この手法を用いることで梶原らの研究[8],[9]では、前節にて説明した造船所 A の実問

題データを用いてジャストインタイム・スケジューリングを達成した。しかしこの手法では同期制約を満足させることができず、一部のラインではサブブロックの完成日より前にサブブロックを必要とするブロックの作業が開始してしまう計画となっていた。

このように、工程計画の制約条件のうち一方を遵守できる計画手法は見受けられても、両条件を満足させるのは難しいのが現状となっている。

1.3.2 本研究のアプローチ

造船所の工程計画問題の定式化から、設備制約と同期制約の遵守を求められている。また、工程計画時の検討内容として製作順序の問題と、設備選択の問題が挙げられる。従来研究について、これらの観点から考察すると、制約条件のどちらか一方は確実に満足させられるが、もう一方がおざなりになりやすい。また製作順序の変更を行うとモデルから組み直しになる場合や、作業する設備を明確にしてない場合が見受けられる。

このような問題を解決するために、設備制約と同期制約を同時に遵守しつつ、製作順序や作業設備の変更が容易に行える計画手法の開発が必要である。そこで本研究室ではHeap 法による逐次処理計画手法を提案した。各作業を、作業設備や人員といった工程の「リソース」として明確にし、そのリソースを使用する時間を用いて、積み木のような「ピース」の形にモデル化する。このピースを積み木のように重ねると、同一のリソースを使う場合は互いが衝突するために、同じタイミングでそのリソースを使用するような設備制約に違反する計画を避けて計画が行える。また積む順番を調整することで、作業の優先順位を考慮でき、これにより同期制約を守った工程計画が行える。

この Heap 法を利用して、実際の造船所のブロック組立計画データを用いて、その実用性の確認を行った。さらに製作順序や製作場所に対して、最適化アルゴリズムである焼きなまし法を用いることで、膨大な組み合わせの中から、作業の行われない無駄な時間を最小にするスケジューリング結果を求めた。

最後に、この Heap 法による逐次処理計画手法を実際の造船所で使用する際に、作業者がより簡単に適用できることを目指し、WEB アプリケーション化を行った。さらに作業者に試作品を評価してもらい、実用性と改善点について意見を求めた。

1.4 本論文の構成

本論文の構成と貢献は以下のとおりである。

第 1 章では、本研究の背景と取り扱う問題について概要を説明する。まず研究の対象とする造船所 A と造船所 B の生産計画の現状について説明する。船舶の設計が終了し、工程計画立案作業へ移行した際に、表計算ソフト上で行う計画作業について述べる。そ

の後、工程計画の問題点についてまとめ、スケジューリング問題としての定式化を行う。定式化された問題に対して、従来法の適用限界について指摘したのち、本研究のアプローチについて説明する。

第 2 章では、本研究にて提案する Heap 法について述べる。まず Heap 法の演算のベースとなる Max-Plus 代数について説明する。次に Heap 法について説明する。Heap 法は横軸に機械や作業区画をリソースとして設定し、縦軸にリソースを使用する時間を設定する。これにより定義される「ピース」を積み上げ計画を行う手法である。このピースにより、設備の許容量を守らせる設備制約と、ある工程に必要な部品が、必要な時間に完成できるようにライン間を同期させる同期制約の問題を解決できる。また本研究ではジャストインタイムを目的とするため、ピースを積み上げず、納期へ引き上げ計画を行う。この Heap 法の動作確認のために、造船所の問題を模したベンチマーク問題を作成し Heap 法を適用した。その結果、設備制約および同期制約の順守を確認した。

第 3 章では、造船所 A の組立工程データを用いて、Heap 法の実問題適用結果について述べる。この造船所の特徴は先行中組ブロックである。先行中組ブロックとは、中組ブロック製作の基礎ブロックで、中組ブロック組立ラインを通過後に再び中組ブロック組立ラインへ搬入される。このためブロックの流れが複雑で工程計画の作成が困難になる。この組立ラインのリソースを定義し、通過するブロックのピースを作成して Heap 法を適用した。その結果、設備制約および同期制約を守り、無駄な待機時間であるアイドル期間を半減させた。またブロックをコンベアへ投入する順序を探索し、さらにアイドル期間を削減する計画を得た。

第 4 章では、造船所 B の組立工程データを用いて、Heap 法の実問題適用結果について述べる。この造船所は工程間にバッファがなく、適切なタイミングで次工程にブロックを送る必要がある。また組立定盤の使用場所を経験で決めており、場合により当日に決定するため、工程計画が最適とは言えない。本研究では使用定盤の決定に最適化手法を用いることで、アイドル期間およびリード期間を短縮した。

第 5 章では、工程計画用 WEB アプリケーションの開発について述べている。造船所では主に表計算ソフトを使用している。反面本研究ではプログラムを用いて計画を作成している。その間を繋ぐため Heap 法による WEB アプリケーションツールを作成し、計画の変更・修正を簡単な操作で行うことを提案し、開発を行った。

第 6 章では結論として、ブロック組立工程におけるジャストインタイム・スケジューリングのために Heap 法の提案と適用したことと、今後の課題について述べている。

第 2 章 Heap 法の提案

2.1 Max-Plus 代数

本研究で提案する Heap 法の数学的基礎となる Max-Plus 代数[10]について説明する。

いま実数の集合を$\mathbb{R}$とする。このとき$\varepsilon = -\infty$、$e = 0$とおく。さらに加法演算および乗法演算を

$$\begin{cases} \mathrm{a} \oplus \mathrm{b} = \max\{a, b\} \\ \mathrm{a} \otimes \mathrm{b} = \mathrm{a} + \mathrm{b} \end{cases} \tag{2.1}$$

と定義した集合$\mathbb{R}_{max} = \mathbb{R} \cup \{\varepsilon\}$を Max-Plus 代数という。このとき$\mathrm{a} \oplus \varepsilon = \mathrm{a}$となるため、εは加法の単位元、$\mathrm{a} \otimes e = \mathrm{a}$となるためeは乗法の単位元となる。乗法の記号⊗は混同のない限りしばしば省略される。

次に、Max-Plus 代数における行列計算について説明する。行列計算は通常の加法演算+を⊕に、乗法演算×を⊗に置き換えておこなわれる。例えば次のように計算される。

$$\begin{bmatrix} \mathrm{a} & \mathrm{b} \\ \mathrm{c} & \mathrm{d} \end{bmatrix} \otimes \begin{bmatrix} e \\ \varepsilon \end{bmatrix} = \begin{bmatrix} \mathrm{a} \otimes e \oplus \mathrm{b} \otimes \varepsilon \\ \mathrm{c} \otimes e \oplus \mathrm{d} \otimes \varepsilon \end{bmatrix} = \begin{bmatrix} \mathrm{a} \oplus \varepsilon \\ \mathrm{c} \oplus \varepsilon \end{bmatrix} = \begin{bmatrix} \mathrm{a} \\ \mathrm{c} \end{bmatrix} \tag{2.2}$$

このとき次の連立 1 次不等式$\mathrm{A} \otimes \mathrm{x} \leq \mathrm{b}$を考える。

$$\underbrace{\begin{bmatrix} \mathrm{a}_{11} & \cdots & \mathrm{a}_{1\mathrm{n}} \\ \vdots & \ddots & \vdots \\ \mathrm{a}_{\mathrm{m}1} & \cdots & \mathrm{a}_{\mathrm{mn}} \end{bmatrix}}_{\mathrm{A}} \otimes \underbrace{\begin{bmatrix} \mathrm{x}_1 \\ \vdots \\ \mathrm{x}_{\mathrm{n}} \end{bmatrix}}_{\mathrm{x}} \leq \underbrace{\begin{bmatrix} \mathrm{b}_1 \\ \vdots \\ \mathrm{b}_{\mathrm{m}} \end{bmatrix}}_{\mathrm{b}} \tag{2.3}$$

このとき$\mathrm{i} = 1, \dots, \mathrm{m}$と$\mathrm{j} = 1, \dots, \mathrm{n}$に対して、$\mathrm{a}_{\mathrm{ij}} \in \mathbb{R}_{max}$、$\mathrm{b}_{\mathrm{j}} \in \mathbb{R}$のとき次が成り立つ。

$$\begin{aligned} & \forall \mathrm{i}: \max\{a_{i1} + x_1, \dots, a_{im} + x_m\} \leq \mathrm{b}_{\mathrm{i}} \\ & \Leftrightarrow \forall \mathrm{i}, \forall \mathrm{j}: \ a_{ij} + x_j \leq \mathrm{b}_{\mathrm{i}} \\ & \Leftrightarrow \forall \mathrm{i}, \forall \mathrm{j}: x_j \leq \mathrm{b}_{\mathrm{i}} - a_{ij} \\ & \Leftrightarrow \forall \mathrm{j}: x_j \leq \min\{\mathrm{b}_1 - a_{1j}, \dots, \mathrm{b}_{\mathrm{m}} - a_{mj}\} \\ & \Leftrightarrow \forall \mathrm{j}: x_j \leq \underbrace{-\max\{a_{1j} - \mathrm{b}_1, \dots, a_{mj} - \mathrm{b}_{\mathrm{m}}\}}_{\mathrm{x}_{\mathrm{j}}^*} \end{aligned} \tag{2.4}$$

この$\mathrm{x}_{\mathrm{j}}^*$を要素に持つベクトル$\mathrm{x}^*$は次式で表される。

$$\mathrm{x}^* = -\mathrm{A}^{\mathrm{T}} \otimes (-\mathrm{b}) \tag{2.5}$$

つまり、この$\mathrm{A} \otimes \mathrm{x} \leq \mathrm{b}$を満足する任意のxに対して$\mathrm{x} \leq \mathrm{x}^*$が成り立つため、$\mathrm{x}^*$は連立 1 次法的式の最大解と呼ばれる。

Heap 法は以上の演算を基礎として考案されている。

2.2 Heap 法

Heap 法[11]とは、横軸に生産機器や人員などのリソースを並べ、縦軸にそのリソースを使用する時間をとることで工程を表す「ピース(piece)」を定義し、そのピースを積み重ねることで計画を行う手法である。ここでは文献にもとづいて、例を挙げつつ Heap 法について説明する。

いま Fig.2.-1 に示す通り、機械 1 を 1 単位時間使用したあとに、機械 1 と機械 2 を同時に 1 単位時間使用する工程 a と、機械 3 を 2 単位時間使用後に機械 2 を 1 単位時間使用する工程 b の 2 つの工程を考える。

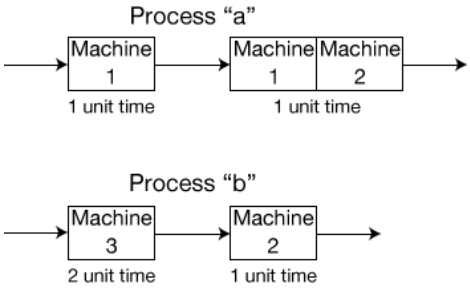

Fig.2.-1 工程 a と工程 b

この 2 つの工程のピースは Fig.2.-2 のように表現できる。ここで各ピースは横軸にリソース番号 1、2、3 をとり、その上に各リソースを使用する時間を示している。つまり工程 a は機械 1 と機械 2 を使用するのでリソースの 1 と 2 を選択し、それぞれ必要な単位時間を設定しており、工程 b についても同様に考えられる。

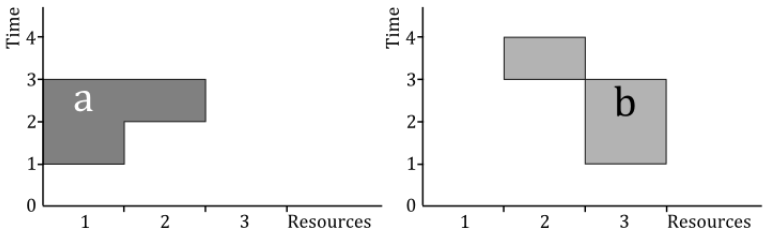

Fig.2.-2 ピース a とピース b

このとき任意のピースηは、使用するリソース番号の集合$\mathbb{R}(\eta)$、作業終了時刻を意味する上側境界線ベクトル$u(\eta)$、作業開始時刻を意味する下側境界線ベクトル$l(\eta)$で構成される。ピース a およびピース b についてこれらを考えると

$$\mathbb{R}(a)=\{1,2\},\quad u(a)=\begin{bmatrix}3\\3\\\varepsilon\end{bmatrix},\quad l(a)=\begin{bmatrix}1\\2\\\varepsilon\end{bmatrix} \tag{2.6}$$

$$\mathbb{R}(b)=\{2,3\},\quad u(b)=\begin{bmatrix}\varepsilon\\4\\3\end{bmatrix},\quad l(b)=\begin{bmatrix}\varepsilon\\3\\1\end{bmatrix} \tag{2.7}$$

となる。

このときピースηに対して、Max-Plus 代数表現された Heap 行列$M(\eta)$の(i,j)要素を

$$\mathrm{M}(a)=\begin{bmatrix}2 & 1 & \varepsilon\\ 2 & 1 & \varepsilon\\ \varepsilon & \varepsilon & e\end{bmatrix} \tag{2.8}$$

$$\mathrm{M}(b)=\begin{bmatrix}e & \varepsilon & \varepsilon\\ \varepsilon & 1 & 3\\ \varepsilon & e & 2\end{bmatrix} \tag{2.9}$$

となる。この Heap 行列が、数値計算においてピースを意味する。

ここで次の式のように、下型境界線ベクトル$l(\eta)$に Heap 行列$M(\eta)$を Max-Plus 代数演算によって掛け合わせると、上側境界線ベクトル$u(\eta)$が得られることに注意する。

$$\underbrace{\begin{bmatrix}2 & 1 & \varepsilon\\ 2 & 1 & \varepsilon\\ \varepsilon & \varepsilon & e\end{bmatrix}}_{M(a)}\otimes\underbrace{\begin{bmatrix}1\\ 2\\ \varepsilon\end{bmatrix}}_{l(a)}=\underbrace{\begin{bmatrix}3\\ 3\\ \varepsilon\end{bmatrix}}_{u(a)} \tag{2.10}$$

$$\underbrace{\begin{bmatrix}e & \varepsilon & \varepsilon\\ \varepsilon & 1 & 3\\ \varepsilon & e & 2\end{bmatrix}}_{M(b)}\otimes\underbrace{\begin{bmatrix}\varepsilon\\ 3\\ 1\end{bmatrix}}_{l(b)}=\underbrace{\begin{bmatrix}\varepsilon\\ 4\\ 3\end{bmatrix}}_{u(b)} \tag{2.11}$$

一般に幾つかの工程を表すピースが集まったものを Heap モデルと呼ぶ。ここで定義したようなピースを生産順序に合うような Heap モデルにすることで、工程計画を行うものが Heap 法による計画手法となる。ピースを積み上げた場合はプッシュ型スケジューリングとなり、ピースを納期へ向けて引き上げた場合はプル型スケジューリングとなる。

本研究ではこの Heap モデルを用いて、リソースの制約を厳守したスケジューリングを行う。

2.3 プッシュ型スケジューリング

プッシュ型スケジューリングとは生産工程の上流から下流へ向かって押し出すように次々と製品を製作するように計画を行う方法である。生産工程の設備が空いていれば製品を製作するため、設備の稼働率は上がるが、次工程の設備で処理できない量の製品が流れこむ計画になる可能性がある。

Heap モデルでは、納期に間に合うようにピースを積み上げていくことでプッシュ型スケジューリングを表現する。例えば Fig.2-4 のようにピース a、b の順に、またはピース b、a の順に積み上げる。ただし両ピースとも機械 2 を使用するが、これはどちらを先に実施しても良いとする。

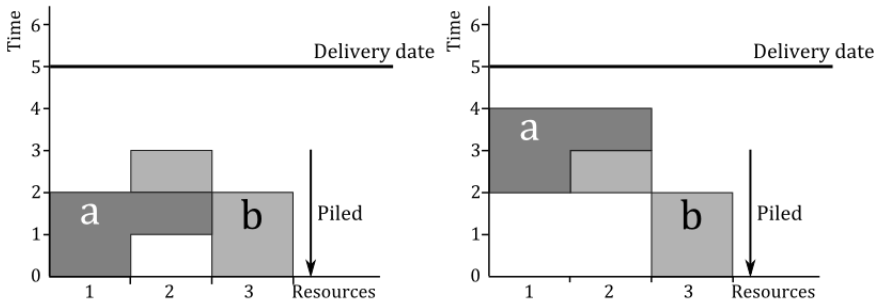

Fig.2-4 ピース a とピース b の積み上げ

Fig.2-4 の左図では、工程 a が機械 2 を先に使用し、その後工程 b が機械 2 を使用する計画となり、右図ではその逆となっている。後者のほうでは終了時刻の遅れがみられ、全体の作業期間であるリードタイムを短くする観点からはふさわしくない計画となる。このようにピースを積み上げる順番が重要となることがわかる。また、終了時刻から納期まで、とくに作業のない時間であるアイドル期間(Idling time)が発生する。この期間は、在庫としてどこかへ保管されることになる。

この Fig.2-4 について、Heap 法による数値計算の例を示す。なにもピースが積み上げられていない初期状態を$\emptyset$としたとき、上側境界線ベクトルは$u(\emptyset) = [\varepsilon \quad \varepsilon \quad \varepsilon]^{\mathrm{T}}$と表すことができる。ここで左図の場合、まずピース a を積み上げるため

$$\underbrace{\begin{bmatrix} 2 & 1 & \varepsilon \\ 2 & 1 & \varepsilon \\ \varepsilon & \varepsilon & e \end{bmatrix}}_{M(a)} \otimes \underbrace{\begin{bmatrix} \varepsilon \\ \varepsilon \\ \varepsilon \end{bmatrix}}_{u(\emptyset)} = \underbrace{\begin{bmatrix} 2 \\ 2 \\ \varepsilon \end{bmatrix}}_{u(a)} \tag{2.12}$$

となり、左図のピース a の上側境界線と一致する。つぎにピース b を積み上げると

$$\underbrace{\begin{bmatrix} e & \varepsilon & \varepsilon \\ \varepsilon & 1 & 3 \\ \varepsilon & e & 2 \end{bmatrix}}_{M(b)} \otimes \underbrace{\begin{bmatrix} 2 \\ 2 \\ \varepsilon \end{bmatrix}}_{u(a)} = \underbrace{\begin{bmatrix} 2 \\ 3 \\ 2 \end{bmatrix}}_{u(ab)} \tag{2.13}$$

となり、左図ピース b を積み上げたあとの全体の上側境界線と一致する。このように積み上げたいピースの Heap 行列を、積み上げる Heap モデルの上側境界線ベクトルと掛け合わせることで新しい上側境界線を得ることで、ピースの積み上げを行う。右図の場合については、積み上げる順番を逆にすることで同様に計算することができる。

Fig.2-4 に対応するガントチャートを Fig.2-5 に示す。ここでジョブ a、b はそれぞれ Fig.2-4 のピース a、b に対応する。

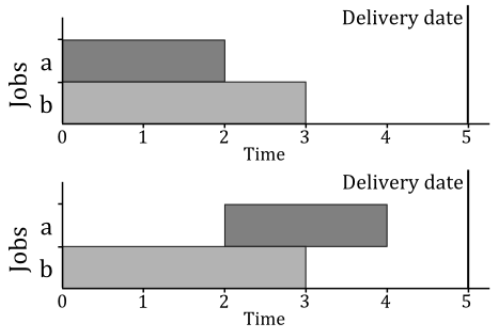

Fig.2-5 Fig.2-4 に示す Heap モデルのガントチャート

2.4 プル型スケジューリング

プル型スケジューリングとは生産工程の下流が必要とするタイミングに完成するように上流に製作指示を送り、製品を引き取るように計画する方法である。製品が次の工程にジャストインタイムで搬入されるため、プッシュ型のように作業待ちによる在庫のあふれが生じない計画となる。

Heap モデルでは、納期へ向けてピースを引き上げることでプル型スケジューリングを表現する。例えば、Fig.2-6 に示すようにピース a、b の順に、またはピース b、a の順に引き上げる。これによりアイドル期間が最小化された計画となる。

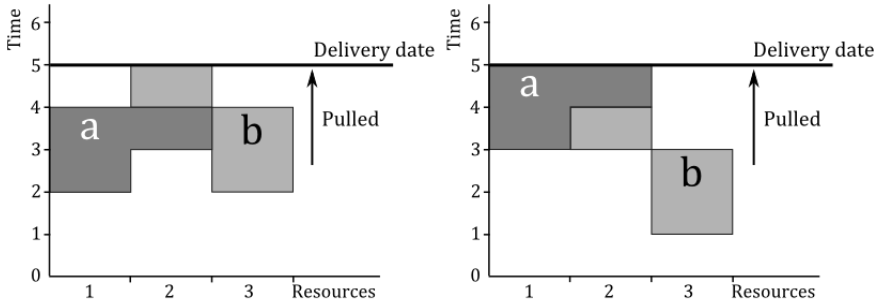

Fig.2-6 ピース a とピース b の引き上げ

このうち左図について、Heap 法による数値計算の例を示す。ピース a、b ともに納期が第 5 日目であるため、なにもピースが引き上げられていない状態$\emptyset$の下側境界線ベクトルは、$l(\emptyset) = [5 \quad 5 \quad 5]^{\mathrm{T}}$と表せる。ピースは b から引き上げられているため、まず$l(\emptyset)$を目標にピース b を引き上げ、納期にジャストインに完了できる開始日$x(b)$を求める。そのためにはピース b に関する次の連立不等式を解く問題として定式化される。

$$\underbrace{\begin{bmatrix} e & \varepsilon & \varepsilon \\ \varepsilon & 1 & 3 \\ \varepsilon & e & 2 \end{bmatrix}}_{M(b)} \otimes \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{x(b)} \leq \underbrace{\begin{bmatrix} 5 \\ 5 \\ 5 \end{bmatrix}}_{l(\emptyset)} \tag{2.14}$$

ここで左辺は作業終了時刻を、右辺は納期を表す。いまこの最大解を求めると(2.5)式より

$$\begin{aligned} x(b) &= -M^{\mathrm{T}}(b) \otimes \left(-l(\emptyset)\right) \\ &= -\begin{bmatrix} e & \varepsilon & \varepsilon \\ \varepsilon & 1 & e \\ \varepsilon & 3 & 2 \end{bmatrix} \otimes \begin{bmatrix} -5 \\ -5 \\ -5 \end{bmatrix} = \begin{bmatrix} 5 \\ 4 \\ 2 \end{bmatrix} \end{aligned} \tag{2.15}$$

となり、ピース b を引き上げた際の下側境界線と一致する。次にこれを新しい目標としてピース a を引き上げる。そのために連立不等式

$$\underbrace{\begin{bmatrix} 2 & 1 & \varepsilon \\ 2 & 1 & \varepsilon \\ \varepsilon & \varepsilon & e \end{bmatrix}}_{M(a)} \otimes \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{x(a)} \leq \underbrace{\begin{bmatrix} 5 \\ 4 \\ 2 \end{bmatrix}}_{l(b)} \tag{2.16}$$

の最大解を求めると

$$\begin{aligned} x(a) &= -M^{\mathrm{T}}(a) \otimes \left(-l(b)\right) \\ &= -\begin{bmatrix} 2 & 2 & \varepsilon \\ 1 & 1 & \varepsilon \\ \varepsilon & \varepsilon & e \end{bmatrix} \otimes \begin{bmatrix} -5 \\ -4 \\ -2 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 2 \end{bmatrix} \end{aligned} \tag{2.17}$$

となり、ピース a、b を引き上げた場合の下側境界線と一致する。右図の場合も引き上げる順を逆にすることで同様の結果を得られる。

以上のように、納期へジャストインするように生産することを目指し、ピースを引き上げることで各工程の開始時刻を決定する計画手法を、Heap モデルによるプル型スケジューリング法、つまり Heap 法として提案する。

2.5 リソース選択規則

プル型スケジューリングでは、アイドル期間の最小化を目的とする。このときリソースの選び方によって、ピースが別のピースに遮られ納期に近づけずアイドル期間が大きくなる。そのため、アイドル期間を最小にするリソースを選択する必要があるが、これは大規模な組合せ最適化問題となるため、短時間で計画を作成したい場合に最適化問題を解くことができない。そこで本研究では、ヒューリスティックな選択規則を提案した。

Fig.2-7 に示すような工程を通過する製品があるとする。この製品は機械 1 を 1 単位時間、機械 2 を 2～4 単位時間使用して製作される。機械 1 は作業区画が 1 区画あり、機械 2 は作業区画が 3 区画ある。

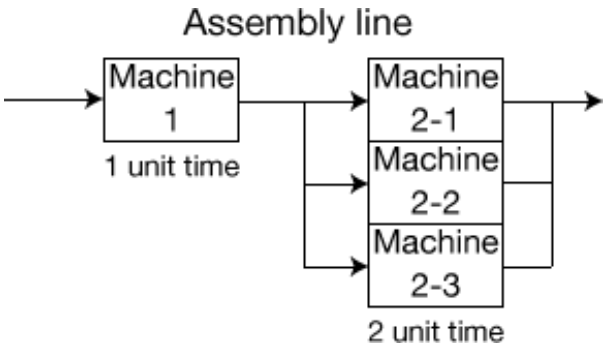

Fig.2-7 組立工程のフロー図

この組立工程を用いて製品を 6 個制作する際の組立工程計画を考える。その製品のデータを Table 2-1 にまとめる。左の列から順に、通し番号、製品名、機械 1 の使用時間、機械 2 の使用時間、納期、製作順序となっている。また工程を通過する製品のピースの例として、Product 1 のピースを Fig.2-8 に示す。このピースでは仮に Product 1 は機械 2 の作業区画 1 番を使用することとしている。

Table 2-1 組立工程を通過する製品のデータ

No.	Name	Machine 1	Machine 2	Due date	Order
1	Product 1	1 unit time	2 unit time	7th day	4
2	Product 2	1 unit time	2 unit time	9th day	6
3	Product 3	1 unit time	2 unit time	9th day	5
4	Product 4	1 unit time	4 unit time	6th day	1
5	Product 5	1 unit time	2 unit time	5th day	2
6	Product 6	1 unit time	2 unit time	6th day	3

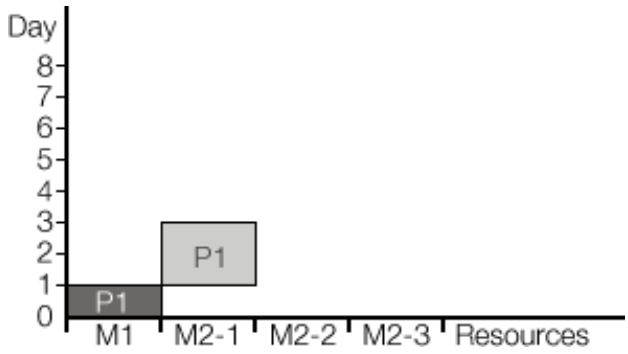

Fig.2-8 Product 1 のピース

さてここで各ピースは、機械 2 のどの作業区画を選択するか考える必要がある。本研究で提案する選択規則について説明する前に、比較対象として 2 つの選択規則を挙げる。

1 つ目は、事前に割り当てられた番号にしたがってリソースを割り当てるものである。Table 2-1 に示す通り、各製品には通し番号が振られている。この割当てられた番号にしたがってリソースを割り当てることを考えると、Product 1 は 1 番であるため、機械 2 の作業区画 1 を使用する。2、3 番の製品も同様に作業区画 2、3 を使用し、4 番の製品はまた作業区画 1 を使用する。最終的に Fig.2-9 に示すような Heap モデルが得られる。このようにリソースを選択するものを Rule 1 とする。

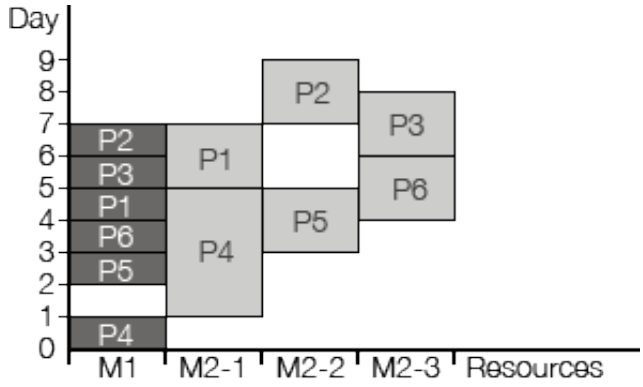

Fig.2-9 Rule 1 によるリソース選択結果

2 つ目は、引き上げられた順に左から順に輪番方式でリソースを割り当てるものである。Table 2-1 に示す通り、製作順序は Product 4, 5, 6, 1, 3, 2 の順となっている。このときプル型スケジューリングの考えに従うと、ピースは 2, 3, 1, 6, 5, 4 と製作順序の逆順に引き上げられることとなる。そのため、引き上げられた順に Product 2 は作業区画 1 を使用し、Product 3 は作業区画 2 を使用する、と輪番で決定していく。最終的に Fig.2-10 に示すような Heap モデルが得られる。これを Rule 2 とする。

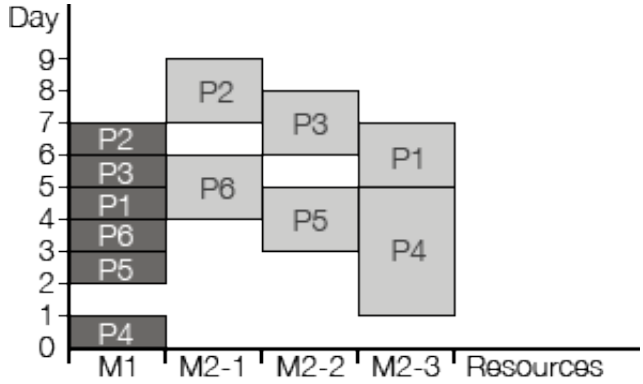

Fig.2-10 Rule 2 によるリソース選択結果

それぞれの Heap モデルを見ると、機械 2 の使用期間に空きがあり、必ずしも効率的に使用されていないことがわかる。また、Product 4 の機械 2 での作業が長く、Product 1 にぶつかりリード期間が伸びていることがわかる。

そこで本研究では、選択規則 Rule 3 として次に示すようなアルゴリズムを提案した。このアルゴリズムは、「アイドル期間を最小にするために納期へ最も近づける区画を選

択したい」と「その区画を出来る限り連続して使用したい」という考えに基づいている。

Algorithm 1: リソース選択規則 rule 3

Step 1：現在の Heap モデルのうち、リソース選択を行いたい範囲の下側境界線ベクトルl_Rと、これから引き上げるピースηの納期$t(\eta)$を得る。

Step 2：$d = l_R - t(\eta)\mathbf{1_n}$を計算する。ここで$d$の要素がすべて負のとき Step 3-1 へ、それ以外のとき Step 3-2 へ進む。

Step 3-1：$i = \arg\min\{|d|\}$となるリソースiを選択し、ピースを引き上げる。

Step 3-2：$i = \arg\min\{|d|\}$かつ$d_i \geq 0$となるリソースiを選択し、ピースを引き上げる

このアルゴリズムを具体例にそって説明する。いま Fig.2-11 に示すように Product 2 および 3 が引き上げられ、次に Product 1 を引き上げる状況を考える。このとき Product 1 の納期(第 7 日目)と現在の Heap モデルにおける機械 2 リソース部分の下側境界線までの差を取ると

$$d = \begin{bmatrix} 7 \\ 6 \\ 9 \end{bmatrix} - 7\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ 2 \end{bmatrix} \tag{2.18}$$

となる。このdの要素について、最小かつ 0 以上のものを選択すると、$d_1 = 0$であるため、Product 1 は機械 2 の作業区画 1 を選択し引き上げることとする。そのため Fig.2-12 にしめすような Heap モデルが得られる。

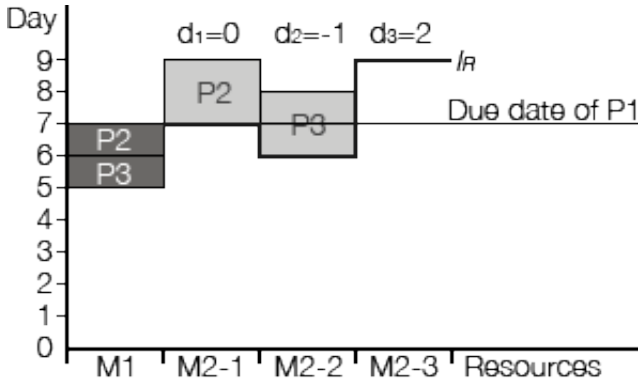

Fig.2-11 機械 2 と Product 1 の納期の差

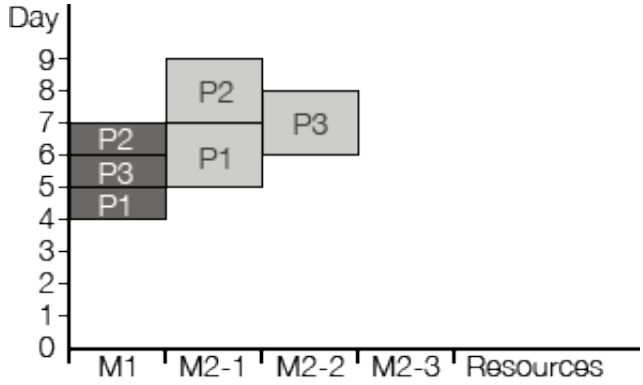

Fig.2-12 Rule 3 によるピース 1 のリソース選択結果

この操作を繰り返し、すべてのピースを引き上げた結果を Fig.2-13 に示す。Rule 1 および Rule 2 と比べるとピース間の隙間が埋まり、設備を連続的に使用する計画となっている。また、ピース P4 のみを作業区画 3 で制作することにより、全体のリード期間を短縮することができている。

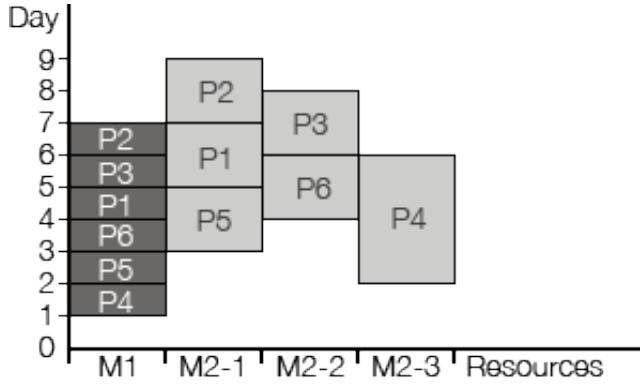

Fig.2-13 Rule 3 によるリソース選択結果

本研究で特に言及しない場合は、リソース選択規則としてこの Rule 3 を使用することとする。

2.6 リソースの畳み込み

Fig.2-14 に示すような組立工程を考える。この組立工程は作業区画を、機械 1 にm区画、機械 2 にn区画、機械 3 にl区画与えられている。作業時間はそれぞれ 3 単位時間とする。例えば各機械の 1 番目の区画を使用した場合のピースは、Fig.2-16 のように定義される。

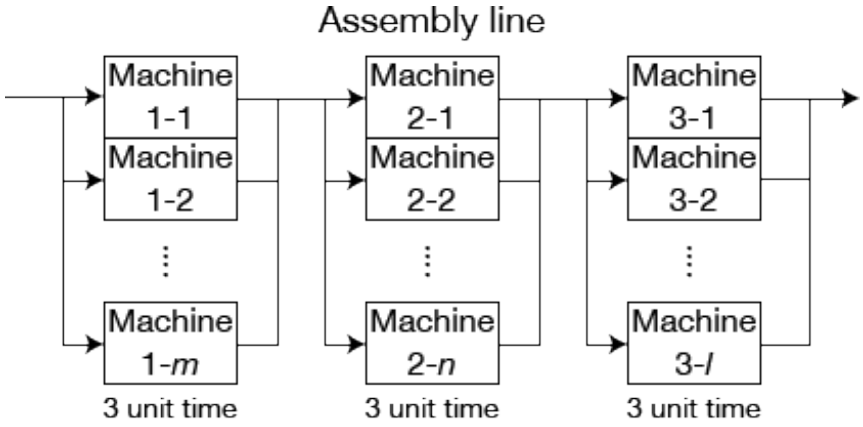

Fig.2-14 組立工程のフロー図

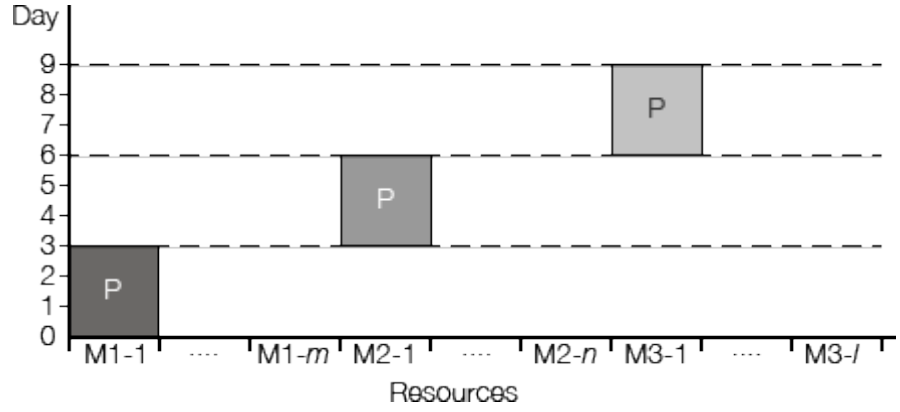

Fig.2-15 組立工程を通過するある製品のピース

このとき全リソース数が$N_{Resource} = m + n + l$であるため、ピースの Heap 行列は $N_{Resource} * N_{Resource}$行列となる。たとえば Fig.2-15 のピースにおいて$m = 3,\ n = 2,\ l = 5$であるとき、その Heap 行列は

$$M(\mathrm{P}) = \begin{bmatrix} 3 & \varepsilon & \varepsilon & e & \varepsilon & -3 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & e & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & e & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 6 & \varepsilon & \varepsilon & 3 & \varepsilon & e & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & e & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 9 & \varepsilon & \varepsilon & 6 & \varepsilon & 3 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & e & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & e & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & e & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & e \end{bmatrix} \tag{2.19}$$

と表現できる。実際の組立工程では、各組立工程の定盤数や、ストックヤードの区画数が大規模になることがあるため、全リソース数$N_{Resource}$が大きくなる。それにより行列のサイズが大きくなるため、数値計算にかかる時間に大きく影響を与えている。

そこで、ここでは同じ機能をもつリソースからひとつ選択する場合に対して、これまで直線的に管理していたリソースにおいて、同一機能のリソースをまとめることで平面的に管理することを提案する。

Fig.2-15 に示すピースは、リソースを直線状に並べて定義している。この時のリソース数を絶対リソース数$N_{absolute} = N_{Resource} = m + n + l$として再定義する。このうち同一の機能をもつものをまとめると、リソースは機械 1～3 の 3 つとなる。この数を相対リソース数$N_{Relative} = 3$と定義する。これは Fig.2-16 に示すように、リソースを縦方向にスタックすることで、見かけ上のリソース数を減らすことを目的としている。

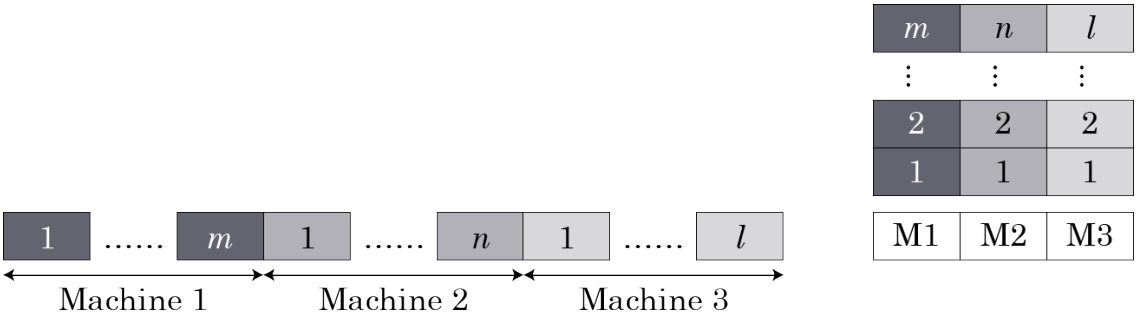

Fig.2-16 現行のリソース定義(左)と提案するリソース定義(右)

これによりピースの定義を行う際に、畳み込んだリソースを用いることで Heap 行列の縮小を行える。(2.19)式に示した Heap 行列は、畳み込みによって

$$M'(\mathrm{P}) = \begin{bmatrix} 3 & e & -3 \\ 6 & 3 & e \\ 9 & 6 & 3 \end{bmatrix} \tag{2.20}$$

と小さくなる。このときのピースを Fig.2-17 に示す。

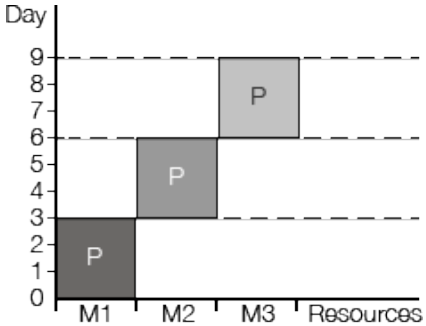

Fig.2-17 提案するリソースに基づくピース

計画を行う際は、同一機能内の使用するリソースを切り替えることで、ピースを引き上げるターゲットである下側境界線を生成する。たとえば機械 1 の作業区画 1、機械 2 の作業区画 1、機械 3 の作業区画 2 を使用する場合は、それぞれの下側境界線を取り出し、順に並べることとなる。

このリソースの畳み込みのもうひとつの利点として、Heap 行列の事前準備ができる点がある。Heap 行列を生成するためには、上側境界線、下側境界線、使用するリソー

スの番号が必要となる。3.5 節にて説明したリソース選択規則 Rule 3 を用いる場合、使用リソースはピースの引き上げ直前に決定される。このため計算直前に Heap 行列を生成するか、事前にすべてのパターンの Heap 行列を用意する必要がある。

直前に生成する場合、計画を 1 案のみ作成する場合は計算コストが低いが、組合せ最適化問題を解く場合、計算コストが高くなり計算時間に大きな影響を与える。事前に宇全てのパターンを用意する場合、絶対リソース数$N_{absolute}$の数が小さい時はピースのパターンは少ないが、$N_{absolute}$が大きくなるとパターンが爆発的に増加するため、事前に用意することが難しくなる。

そこでリソースの畳み込むことで、ピースではなくターゲットとなる下側境界線を変更するため、使用するリソースに依存しない形で Heap 行列を計算することができる。これは畳み込んだリソースにすべてのパターンのピースを含んでいる状態にあると考えられる。

本研究ではこの畳み込んだリソース表現によって Heap 行列を作成することとする。

2.7 ベンチマーク問題の概要

ここまで説明した Heap 法が、組立工程計画問題に対して、狙い通りの効果を発揮するか確認する。そのために、まず実際の工程を模したベンチマーク問題を作成し、Heap 法を適用した結果を調査する。

Fig.2-18 に、造船所の組立工程を模したベンチマーク問題を示す。

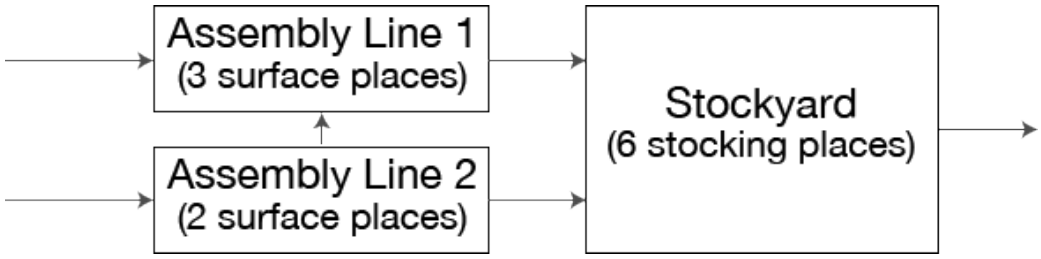

Fig.2-18 ベンチマーク問題で扱う組立工程のレイアウト

この組立ラインは、ブロックの組立に固定定盤を使用している。ここでの作業が終わるとストックヤードに蔵置され、簡単な艤装作業が行われる。その後搭載開始日になると、ドックへ搬入される。このうち一部のブロックは組立ライン 2 で組み立てられた後、組立ライン 1 でさらに大きなブロックとなる。組立ライン 1 は固定定盤を 3 定盤、ライン 2 は固定定盤を 2 定盤備えている。ストックヤードはブロックを最大 6 区画持っており、各ラインにそれぞれ 3 区画ずつ割り当てる。

ここでは各ラインとそれに割り当てられたストックヤードをひとつの工程とみなして Heap 法でモデル化する。以下、組立ライン 1 とそのストックヤードを工程 1、組立

ライン 2 とそのストックヤードを工程 2 と呼ぶ。このベンチマーク問題では、設備の制約を守りつつ、工程 1 と工程 2 間の同期ができた生産計画を立てる必要がある。

この工程を通過するブロックのデータを Table 2-2 に示す。

Table 2-2 ベンチマーク問題のブロックデータ

No.	Name	Line	Work time	Due date	Link
1	S1-B1	ST 1	3	6	0
		AL 1	2		
2	S1-B2	ST 1	2	6	0
		AL 1	2		
3	S1-B3	ST 1	2	9	0
		AL 1	2		
4	S1-B4	ST 1	2	9	0
		AL 1	1		
5	S1-B5	ST 1	3	12	0
		AL 1	2		
6	S1-B6	ST 1	3	12	0
		AL 1	3		
7	S1-B6-P	ST 2	2	12	6
		AL 2	2		
8	S2-B1	ST 2	2	7	0
		AL 2	2		
9	S2-B2	ST 2	1	7	0
		AL 2	2		
10	S2-B3	ST 2	3	10	0
		AL 2	2		
11	S2-B4	ST 2	1	10	0
		AL 2	3		
12	S2-B5	ST 2	3	13	0
		AL 2	2		
13	S2-B6	ST 2	2	13	0
		AL 2	2		

Table 2-2 は左から、番号、名前、作業するライン名、作業時間、納期、次の作業番号となっている。例として、S1-B6-P をみると、このブロックは工程 2 で組み立てられるブロックで、組立ライン 2 で 2 日間、ストックヤードで 2 日間作業が行われる。このブロックは S1-B6 のベースとなるブロックで、工程 2 での作業が終わると、工程 1 へ送られ S1-B6 として更に作業が加えられる。最終的に納期 12 日目に完成すれば良い。

2.8 Heap 法によるプル型スケジューリングの適用

Heap 法をベンチマーク問題に対して適用し、アイドル期間が削減できているかどうか確認する。

工程 1、工程 2 ともにリソースは固定定盤とストック区画になる。仮に固定定盤での作業期間を S 日、ストックヤードでの艤装期間を T 日とすると、Fig.2-19 に示すようなピースを定義する事ができる。

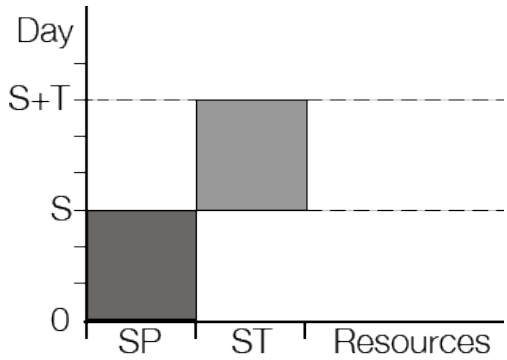

Fig.2-19 ベンチマーク問題で扱うブロック組立を表すピース

固定定盤のある定盤上で S 日間作業されるため、まずリソース SP(Surface Plate)に S 日間存在するように定義する。その後ストックヤードにて T 日間作業が行われるため、リソース ST(Stockyard)に T 日間存在するように定義する。両者の間にバッファは存在せず、作業完了後すぐに定盤からストックへ移動するため、ピース間に隙間はなく連続している。そのためこのピースの全体の作業時間は S+T 日になる。このようなピースをブロックごとに定義し、それらを製作順序にしたがって引き上げる。ここで製作順序は現行計画の開始日順を利用する。

ベンチマーク問題の現行計画について工程 1 の Heap モデル図を Fig.2-20 に、ガントチャートを Fig.2-21 に示す。また工程 2 の Heap モデル図を Fig.2-22 に、ガントチャートを Fig.2-23 に示す。工程 1 ではアイドル期間が 8 日間、工程 2 ではアイドル期間が 13 日間存在する。この工程計画に対して、Heap 法によるプル型スケジューリングを適用する。

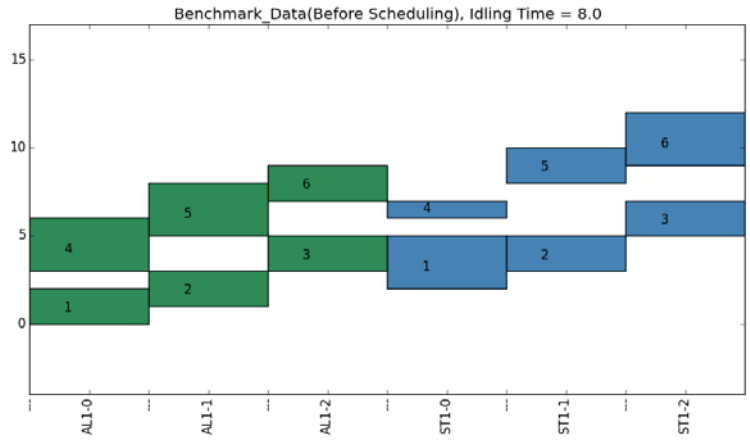

Fig.2-20 ベンチマーク問題工程 1 の現行計画の Heap モデル図

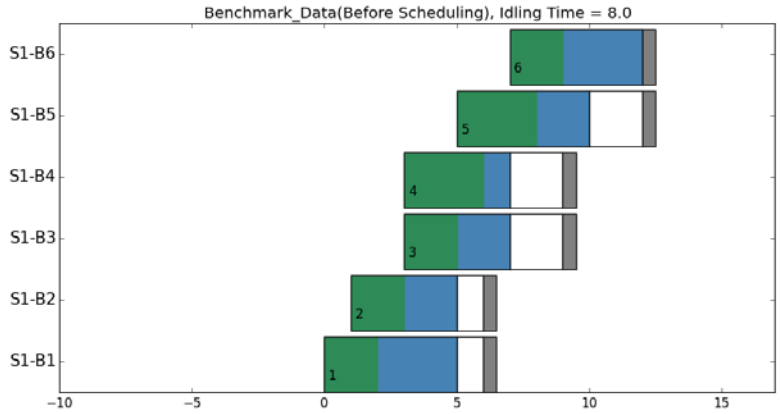

Fig.2-21 ベンチマーク問題工程 1 の現行計画のガントチャート

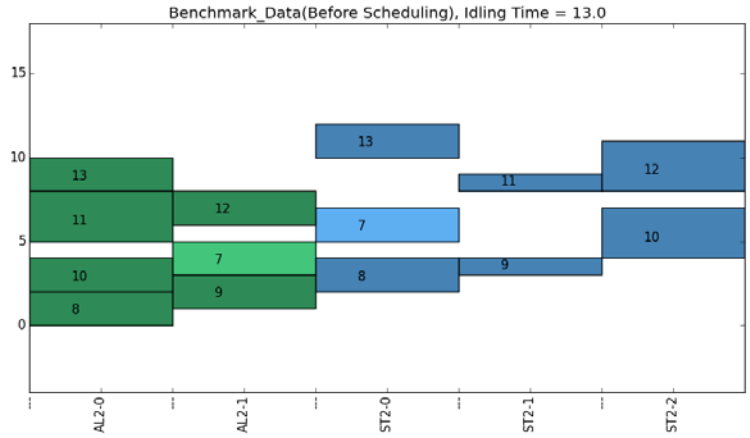

Fig.2-22 ベンチマーク問題工程 2 の現行計画の Heap モデル図

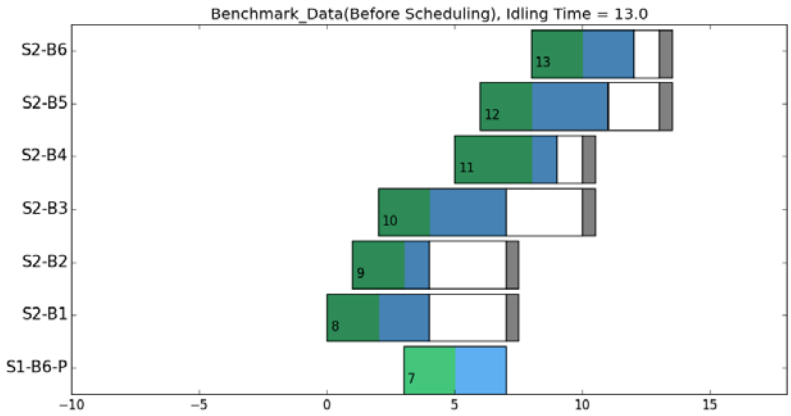

Fig.2-23 ベンチマーク問題工程 2 の現行計画のガントチャート

まず現行計画と同じ作業区画を使用したままで、プル型スケジューリングを適用した結果を示す。工程 1 の適用後 Heap モデル図を Fig.2-24 に、ガントチャートを Fig.2-25 に示す。また工程 2 の適用後 Heap モデル図を Fig.2-26 に、ガントチャートを Fig.2-27 に示す。Heap 法適用によって、工程 1 のアイドル期間は 0 日に、工程 2 のアイドル期間は 6 日に減少した。

工程 1 についてはピースが互いに障害とならずに、納期へ向けて引き上げることができたために、アイドル期間が 0 日となった。Heap モデル図をみると、連続稼働する計画よりも間に大きく隙間時間が生まれるようになっており、設備の使用効率からみると悪い結果となっている。

工程 2 についてはブロック 8~10 番のピースが、自身より上にあるピースに引っかかり納期に近づくことができずにいる。そのため作業を行う区画を変更することがアイドル期間削減の要点となっていることがわかる。

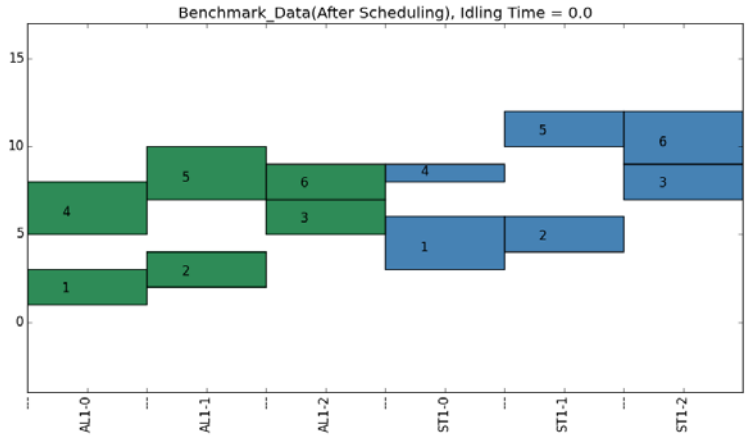

Fig.2-24 ベンチマーク問題工程 1 の適用後 Heap モデル図

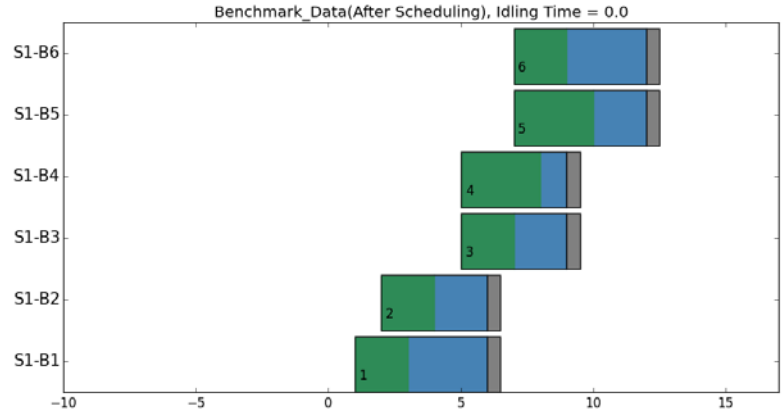

Fig.2-25 ベンチマーク問題工程 1 の適用後ガントチャート

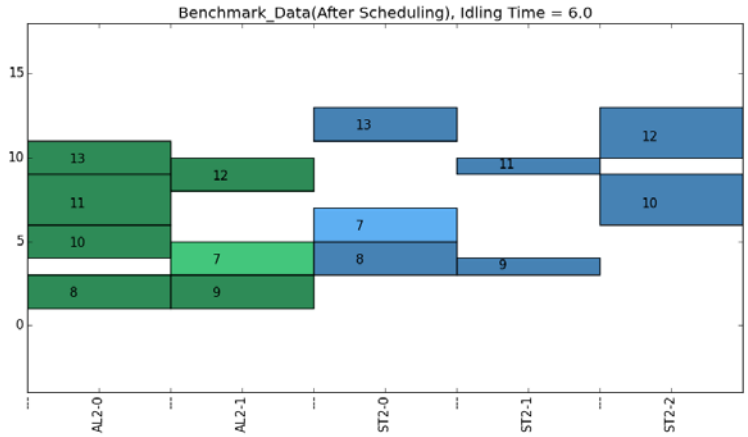

Fig.2-26 ベンチマーク問題工程 2 の適用後 Heap モデル図

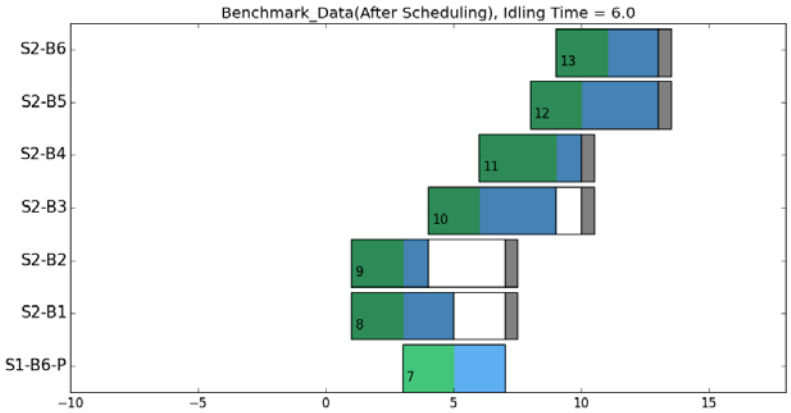

Fig.2-27 ベンチマーク問題工程 2 の適用後ガントチャート

次に、作業を行う定盤・区画を計画中に適宜変更するリソース選択規則 Rule 3 を用いて、ベンチマーク問題のプル型スケジューリング適用を行った結果を示す。

工程 1 の適用後 Heap モデル図を Fig.2-28 に、ガントチャートを Fig.2-29 に示す。また工程 2 の適用後 Heap モデル図を Fig.2-30 に、ガントチャートを Fig.2-31 に示す。Heap 法適用によって、工程 1 のアイドル期間は 0 日に、工程 2 のアイドル期間は 3 日に減少した。

工程 1 はリソース選択規則 Rule 3 によってストックヤードの使用区画数が 2 区画に節約され、各区画が連続的に使用される計画となった。この結果によって、工程 1 ではさらにブロックを組み立てる余裕が設備にあることが明確になった。

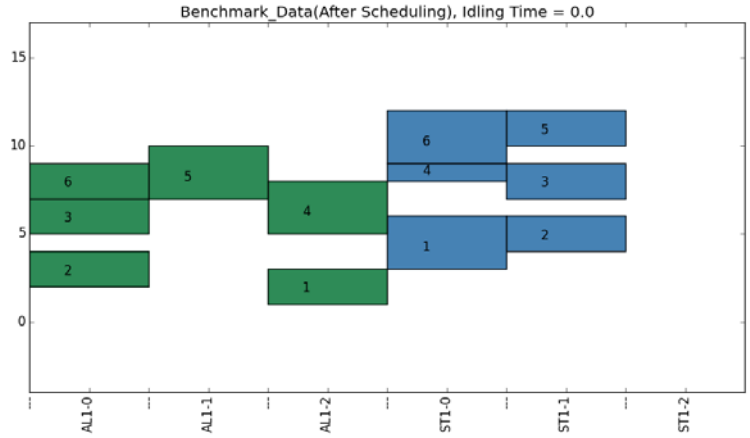

Fig.2-24 ベンチマーク問題工程 1 の適用後 Heap モデル図

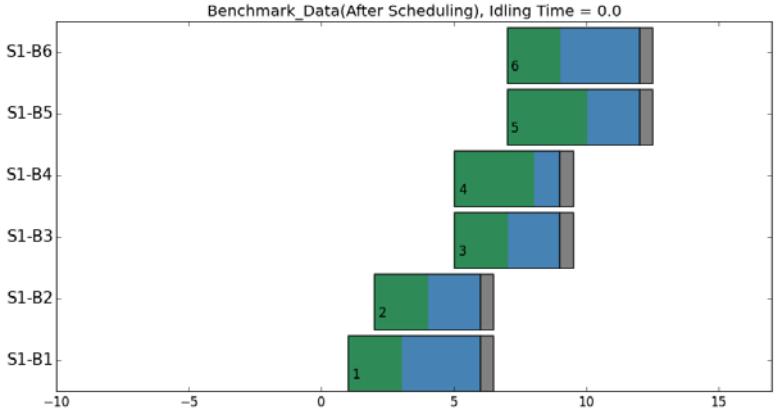

Fig.2-25 ベンチマーク問題工程 1 の適用後ガントチャート

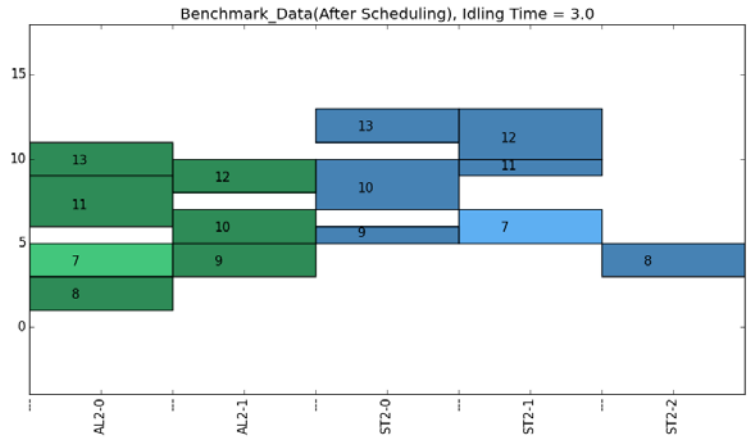

Fig.2-26 ベンチマーク問題工程 2 の適用後 Heap モデル図

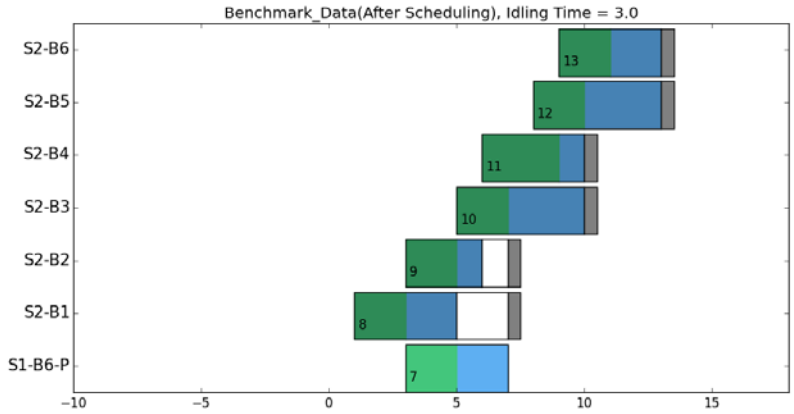

Fig.2-27 ベンチマーク問題工程 2 の適用後ガントチャート

工程 2 はリソース選択規則 Rule 3 によって組立ライン 2 とストックヤードの整理が行われ、以前の結果に比べてよりアイドル期間を削減することができている。また先行作業が行われるブロック番号 7 の S1-B6-P に注目すると、作業が終わる日が工程 1 のブロック番号 6 の開始日と一致するように計画が作成されており、同期制約を満たせていることが確認できた。

このようにベンチマーク問題によって、工程の設備制約と同期制約を満たした状態で、計画を作成することができることを確認した。

2.9 焼きなまし法

Heap 法を用いることで、設備制約および同期制約を自動的に満足させることができる。これにより製作順序とリソースの選択を計画作成者が決定することで、納期にジャストインタイムとなる工程計画をコンピュータにより生成できる。

ジャストインタイムを達成する際に最小化すべき目的関数であるアイドル期間は、製作順序やリソース選択に依存している。そのため、よりアイドル期間を小さくする工程計画を得るためには、膨大な組合せとなる製作順序最適化問題、リソース選択最適化問題を探索する必要がある。本研究ではこの組合せ最適化問題に対して、有力なメタ戦略である焼きなまし法(Simulated Annealing, SA)[12]を用いる。

焼きなまし法は、焼きなましと呼ばれる加熱された個体の冷却過程をシミュレートするアルゴリズムである。温度に依存した確率に応じて状態を推移させるため、温度が高いうちは改悪の状態への推移確率が高くなるが、温度が低くなるにつれ改善の方向に推移する。これにより局所解に陥りにくい特徴がある。

多くの最適化解法が局所解に陥りぬけ出すことが出来にくい欠点を持つが、焼きなまし法は理論上では真の最適解に、実用上では準最適解に到達できる頑強性をもっている。目的関数に対する制約もほぼなく、柔軟性をもった設定が行える利点もある。反面、最適解を得るために非常に多くの計算コストが必要となるため、計算量によっては並列計算などの高速化が求められる。今回の研究では、組み合わせ数が膨大であるものの、ひとつの解候補の計算コストが小さいため、一般的なコンピュータ 1 台で計算を行っている。焼きなまし法アルゴリズムを Algorithm 1 に示す。

Algorithm 2: 焼きなまし法 (Simulated Annealing)

Step 0: 初期化

初期状態$\boldsymbol{x}$の生成、ステップ数$k \coloneqq 0$、温度$T_k \coloneqq T_{ini}$

Step 1: アニーリング

状態$\boldsymbol{x}$の近傍から候補$\boldsymbol{x}'$を生成。目的関数より$z(\boldsymbol{x})$、$z(\boldsymbol{x}')$を得る。差分$\Delta z = z(\boldsymbol{x}') - z(\boldsymbol{x})$を計算し、受理関数$Pr(\Delta z, T_k)$によって新しい状態$\boldsymbol{x}'$を受理するか判定する。受理する場合は$\boldsymbol{x} \coloneqq \boldsymbol{x}'$とする。一定回数 Step 1 を繰り返す。

Step 2: クーリング

十分 Step 2 を繰り返した後クーリング処理を行う。$k = k + 1$かつ$T_{k+1} = F(T_k)$とする。温度が停止条件に達していない場合 Step 2 へ戻る。達している場合 Step 3 へ進む

Step 3: 終了

温度が十分に下がったとき、その時の$\boldsymbol{x}$を最適状態、$z(\boldsymbol{x})$を最適値として終了する。

焼きなまし法を適用するためには、まず初期状態を設定する必要がある。この初期状態は、製作順序最適化問題・リソース選択最適化問題ともに現行計画の時点での製作順序やリソース選択(定盤計画)を用いる。Step 1 のアニーリング処理では、現在の解から近傍の解を生成して、目的関数を計算する。このとき近傍解の生成する方法についていくつかの処理が考えられるが、製作順序の場合にはランダムに選んだ 2 つのピースについて製作順序を入れ替えることで近傍解を生成する。リソース選択の場合には、検討の対象となる工程を通過するピースのうち 1 つをランダムに選択し、同じ工程内の隣接する定盤へ移動させることで近傍解を生成する。近傍解の目的関数を計算したのち、その解を受け入れるかどうか、受理関数にて判断する。この受理関数は

$$\Pr(\Delta z, T_k) = \begin{cases} 1 & if \ \ \Delta z < 0 \\ \exp\left(\dfrac{-\Delta z}{T_k}\right) & otherwise \end{cases} \tag{2.21}$$

を用いた。一連の処理を一定回数繰り返す必要があるが、この回数は 100 回とした。

Step 2 ではクーリング処理を行う。現在の温度をある比率に従って冷却する必要がある。この冷却関数$T_{k+1} = F(T_k)$について、最適解への漸近収束性を保証するために

$$F(T_k) = \frac{T_1}{\log k} \tag{2.21}$$

以上に急速に冷却してはならない。しかしこの冷却関数に従った場合計算速度があまりにも遅いために、現実的な時間で運用することができない。そのため参考文献では、

$$F(T_k) = \lambda T_k \tag{2.22}$$

を推奨している。$0.8 \leq \lambda < 1$の範囲で決定する必要があるが本研究では$\lambda = 0.9$とする。また(2.22)式では冷却が早過ぎると判断される場合、

$$F(T_k) = \frac{T_1}{\sqrt{k}} \tag{2.21}$$

のように対数ではなく平方根を用いた冷却関数を用いることも考えられる。どちらの冷却関数を使用するかについては適宜指定する。本研究では、このアルゴリズムを基にプログラムを作成した。

2.10 考察

第 3 章では、工程計画手法の提案として、Heap 法の数学的な背景と理論を説明した。その後ベンチマーク問題を用いて、設備制約問題、同期制約問題を満たしていることを確認した。またベンチマーク問題においてリソース選択規則 Rule 3 を用いることで、リソースをより効率的に使用しながら工程計画を行うことができることが確認できた。また、リソースの使い方がアイドル期間の削減や、使用する設備の許容量の節約に影響

を与えることを明らかにした。

さらに、製作順序最適化問題やリソース選択最適化問題に備え、最適化アルゴリズムである焼きなまし法について説明をおこなった。

第 3 章 造船所 A の組立工程計画検討

3.1 造船所 A の概要

Fig.3-1 にある造船所 A の組立工程のレイアウトを示す。

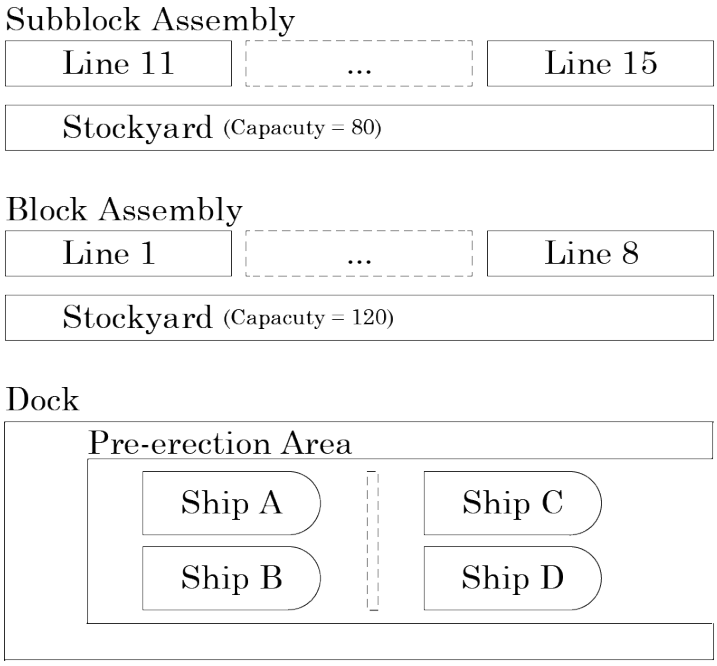

Fig.3-1 造船所 A の組立工程レイアウト

ブロック組立工程の上流から下流に向かって、中組ブロック組立ライン、ストックヤード、大組ブロック組立ライン、ストックヤード、総組場を経て、ドックへ至るのが、主なブロック制作の流れである。このうち一部の中組ブロックについては、この造船所の特徴である先行中組ブロックが混在しているため、中組ブロック組立ラインを通過後、ストックヤードにて先行艤装が行われた後、再び中組ブロック組立ラインへ流される。

この造船所ではツインタンデム建造を行っているため、通常の造船所と比較して、組立工程内に流れるブロック数が多い。そのため、搭載のタイミングにあわせるようにブロックを完成させなければ、ストックヤードのあふれを招く。また上記のように、先行中組ブロックが存在するため、先行中組と通常中組の製作開始日と完了日をあわせるために、同一ラインもしくは複数ライン間での同期をとる必要がある。

今回の検討では、大組ブロック組立ラインのうち 8 ライン、中組ブロック組立ラインのうち 5 ラインに注目し、個別にモデルを立てて工程計画を作成する。それぞれのラインに対して、ストックヤードの区画を割り当てる。これらのパラメータを Table 3-1 および Table 3-2 にまとめる。この造船所の組立ラインはコンベアを用いているため、コンベア定盤数、定盤の移動ピッチのパラメータがある。また各ラインに対して割り当て

たストックヤードの区画数を記載している。

Table 3-1 大組工程ラインのパラメータ

conveyor	m	c	stockyard	n
Line 1	6	7/8	Stock 1	21
Line 2	1	7/8	Stock 2	4
Line 3	1	7/8	Stock 3	4
Line 4	10	7/10	Stock 4	34
Line 5	4	6/4	Stock 5	10
Line 6	9	4.5/9	Stock 6	22
Line 7	7	5/7	Stock 7	15
Line 8	4	4/4	Stock 8	10

m: number of surface plates on conveyor

c: pitch of conveyor (days divided by plates)

n: number of stock areas assigned to conveyors

Table 3-2 中組工程ラインのパラメータ

conveyor	m	c	stockyard	n
Line 11	8	4/8	Stock 11	10
Line 12	24	7/24	Stock 12	31
Line 13	10	4/10	Stock 13	13
Line 14	9	4/9	Stock 14	12
Line 15	10	4/10	Stock 15	14

m: number of surface plates on conveyor

c: pitch of conveyor (days divided by plates)

n: number of stock areas assigned to conveyors

また中組ラインでのブロックの流れを図示したものを Fig.3-2 に示す。ライン 11、12、14 はそれぞれ独立しており、完成したブロックは大組ブロックでの作業へと移る。しかしライン 13 および 15 については、先行中組ブロックが存在するため、完成したブロックの一部は自ラインもしくは他ラインへ搬入され中組ブロック作製の基礎となる。そのため、スケジューリングの順番としては、まず大組ブロックの 8 ラインの生産計画

を行った後、その開始日を目標日として、中組工程のライン 11、12、14 を計画する。その後、自ラインのみで先行中組が完結しているライン 15 の計画を行い、ライン 13 で行われる先行中組ブロックの目標日を決定する。最後に、大組開始日と中組開始日を目標日にライン 13 のスケジューリングを行う。

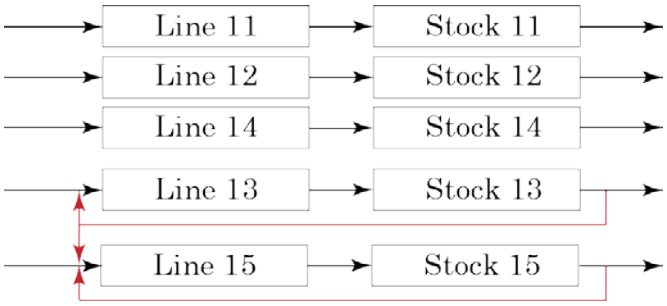

Fig.3-2 中組工程ラインのライン間接続フロー図

3.2 組立工程の Heap モデル

Heap 法を適用するために、まず組立工程のリソースを定義して、工程を通過するブロックのピースを検討する。

Fig.3-3 に定盤数m、通過日数t、コンベアピッチt/mのコンベアラインの模式図を示す。ここで、$CV_1, \ldots, CV_m$は定盤を、$P_1, \ldots, P_m$は定盤を置く区画を表す。各ブロックはまずP_1に置いた定盤上で組立が開始される。その定盤はコンベアピッチごとに$P_1, P_2, \ldots$と移動していき、t日が経過するとP_mにたどり着き、定盤上のブロック組立作業が完了する。

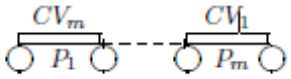

Fig.3-2 コンベアラインの模式図

このコンベアラインを通過するブロックのピースを考える。このコンベアラインを表すためには、「ブロックがどの定盤を占有して組み立てられるのか」という定盤占有情報と、「ブロックの載っている定盤が現在どの区画にいるのか」という定盤位置情報を管理する必要がある。本研究では、このふたつの情報をリソースとして扱うことで、Fig.3-3 のようなピースを作成した。

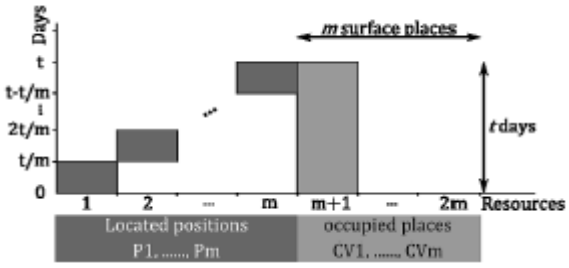

Fig.3-3 コンベアを通過するブロックのピース

横軸のリソース番号は、定盤位置情報を表す$\{1,2,\cdots,m\}$、定盤占有情報を表す$\{m+1,m+2,\cdots,2m\}$である。ピースのうち定盤位置情報を表す部分に配置された階段状の長方形のピースは、ブロックを組み立てる定盤がコンベアラインの始端(P_1)から終端(P_m)までコンベアピッチごとに移動する様子を表している。定盤占有情報を表す部分に配置された縦長の長方形のピースは、ブロックがどの定盤を占有しているかを表すため、リソースのどれかひとつを選択し、高さt日の長方形を置く。Fig.3-3 の例では、ブロックは第 1 番目の定盤で組み立てられることを意味している。

次に、コンベアラインからストックヤードに移されたブロックのピースを考える。Fig.3-4 にストック区画をn区画持つストックヤードの模式図を示す。

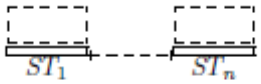

Fig.3-4 ストックヤードの模式図

ストックヤードはブロックを蔵置する区画として$ST_1, ST_2, \ldots, ST_n$をもつ。これらはコンベアと違い移動しないため、「ブロックがどの区画を占有しているのか」だけを管理すれば良い。このストックヤードのうち区画ST_1にブロックをs日間蔵置する場合のピースを Fig.3-5 に示す。

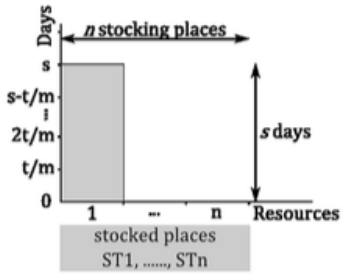

Fig.3-5 ストックヤードに蔵置中のピース

このピースにおけるリソース番号は、区画占有情報を表す$\{1,\ldots,n\}$である。区画ST_1を選択したため、リソース番号 1 番にs日の高さをもつ長方形を配置したピースとなってい

る。

以上の 2 つのピースを組み合わせて、ブロック組立工程を通過するブロックのピースを作成した。これを Fig.3-6 に示す。ただしストックヤード区画のリソース番号は $\{2m+1, \ldots, 2m+n\}$としている。このピースは、コンベアラインでt日間作業したあと、間を開けずにストックヤードに蔵置される工程を表している。ブロックごとに使用する定盤やストック区画は変化することに注意する必要がある。

この組立工程モデルのうち、定盤位置情報を表す部分に注目する。コンベアラインの始端に定盤があるタイミングがわかれば、コンベアピッチによりどの位置に定盤がいるか推測することができる。そのため、リソース 1 番のコンベア始端以外を省略することでピースの簡略化が行える。このピースを Fig.3-7 に示す。

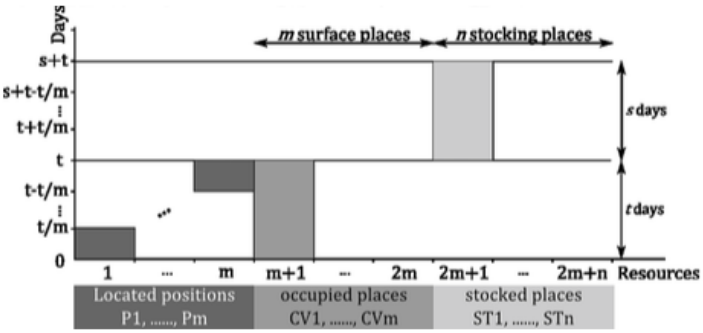

Fig.3-6 組立工程を通過するブロックのピース

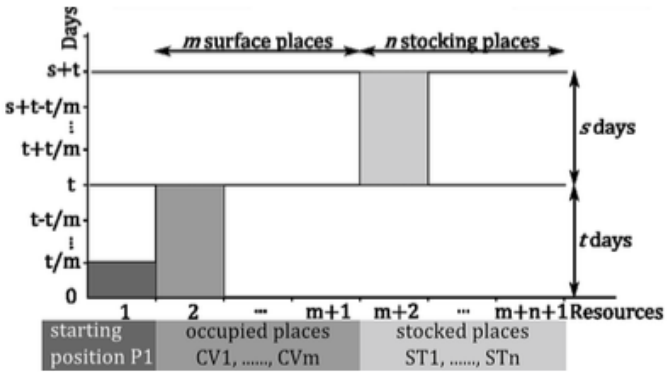

Fig.3-7 ピースの簡略表示

このピースを、各工程を通過するピースひとつひとつに対して作成する。

対象造船所のブロックデータの一部を Table 3-3 に示す。対象造船所から提供されたブロックデータは、左の列から、データ番号、ブロック名、作業日数、最小作業日数、作業定盤、開始日、終了日、工程 ID、接続となっている。ブロック名は本来のブロック名を通し番号に置き換えている。作業日数のうち、作業が効率的に進んだ場合の日数

を最小作業日数としている。作業定盤はそのブロックに対する作業を行う場所を示していて、組立工程については Table 3-1 や Table 3-2 に示す名前を使用している。それ以外の作業については総組場(PE)と小組工程(SA)の対応するものを入力している。開始日および終了日は現行計画のものである。このうち総組作業の開始日が大組の納期であるとみなして計画を行う。工程 ID は各工程に番号を振ったもので、0:総組、1:大組ストック、2:大組ライン、3:中組ストック、4:中組ライン、 5:先行中組ライン、6:小組ラインに対応している。接続は次に行う作業の番号を示していて、例えば作業 7 番の小組作業が終わると作業 6 番を行うということを意味する。

Table 3-3 ブロックデータの例

No.	Name	WT	MWT	Line	Start	Stop	ID	Link
1	BL1001	1	1	PE	100	100	0	0
2	BL1001	6	4	Stock 1	95	100	1	1
3	BL1001	4	4	Line 1	91	94	2	2
4	BL1001	6	6	SA	83	88	6	3
5	BL1001	3	1	Stock 11	89	91	3	3
6	BL1001	7	7	Line 11	82	88	4	5
7	BL1001	4	4	SA	78	81	6	6

WT: Working Time

MWT: Minimum Working Time

このブロックデータから、今回検討する組立工程に関わるブロックデータを抽出し、コンベア作業期間、最小ストック期間、納期、工程の接続関係などをまとめ、ピースを作成することになる。

3.3 Heap 法の適用

4.1 節にて述べた大組工程 8 ライン、中組工程 5 ラインに対して Heap 法によるプル型スケジューリングを適用した。

この生産計画では、ブロックの製作順序に対象造船所の計画現場にて使用される規則である優先規則を使用している。これは「次工程開始日−最小ストック期間」の早いブロックから製作する、というものである。組立工程はコンベアラインを使用しているために、ブロックごとの組立作業時間に差はない。しかしストックヤードでの先行艤装の

時間はブロックごとに異なるため、1 日でおわるものもあれば数日必要とするものも存在する。目標日とストック期間の差をとることで、目標日丁度に終わった場合のストック開始日を得られる。その日付が早いものから作成することで、無駄なストック期間であるアイドル期間を比較的小さくした計画が作成できる。

組立作業がどのリソースで行われるかは、ピースを引き上げる直前に決定する。コンベアについては、設備の性質上から定盤を輪番方式で使用していく。ストックヤードについては、リソース選択規則 Rule 3 を使用した。

以上の条件のもと得られたスケジューリング結果について述べる。まず大組ブロック組立工程に対する Heap 法適用によるアイドル期間の変化を Table 3-4 に示す。

Table 3-4 大組工程計画によるアイドル期間の変化

	Cycle						
Line	1	2	3	4	5	6	7
Line 1	38.1 (75)	11.1 (58)	9.8 (74)	32.3 (69)	25.1 (73)	32.1 (159)	17.1 (112)
Line 2	6 (4)	4 (8)	4 (20)	8 (15)	2 (10)	1 (7)	3 (12)
Line 3	1 (14)	0 (27)	0 (22)	3 (29)	0 (78)	4 (6)	2 (58)
Line 4	56.5 (128)	43.9 (65)	86.7 (183)	92 (244)	53.3 (176)	33.6 (225)	56.8 (297)
Line 5	34 (129)	14.5 (95)	22 (99)	78.5 (112)	14 (98)	49.5 (199)	41 (156)
Line 6	402.6 (286)	221.6 (232)	450.6 (289)	313.3 (202)	260.6 (187)	203.1 (425)	422.6 (518)
Line 7	268 (458)	173.4 (218)	181.7 (639)	266 (470)	210.1 (425)	288.1 (479)	297.3 (453)
Line 8	35 (112)	24 (152)	12 (94)	18 (84)	14 (136)	53 (278)	30 (161)
Total	873.4 (1206)	496.1 (965)	773 (1430)	829.3 (1235)	587.5 (1183)	673.2 (1778)	875.4 (1767)

表内のカッコ内は現行計画のアイドル期間を意味している。Heap 法適用の結果、ほとんどのラインでアイドル期間の減少を確認できた。Line 1 のサイクル 1 番目、Line 6

のサイクル 1、3、4、5 番目で現行計画に比べてアイドル期間が増加した。これらの結果のうち、Line 4 のサイクル 1 番目、Line 5、Line 6 に注目する。

サイクル 1 の Line 4 のスケジューリングについて、現行計画のガントチャート、Heap 法適用後の Heap モデル図、ガントチャートを Fig.3-8 に示す。ガントチャートの横軸は日数、縦軸はブロックである。緑色の棒がコンベア作業、青色の棒がストック期間、黄色が納期を示していて、ストック期間終了から納期までの間の白の空白が「特に作業の行われていない無駄な時間」であるアイドル期間を意味している。左の現行計画のガントチャートを見ると 37 番目、38 番目のブロックなど大きくアイドル期間が発生しているブロックがある。これらのブロックをピースに変換し、納期へむかって引き上げたのが中央の Heap 図になる。この Line 4 にはストック区画を 34 区画割り当てていた。しかしリソース選択規則 Rule 3 による区画選択によって、15 区画までリソースを節約して計画することができている。この Heap モデル図をガントチャートへ変換したものが右図になる。現行計画のガントチャートと比べると、目立っていたアイドル期間を削減した出来る限り納期にジャストインの計画を作成できている。

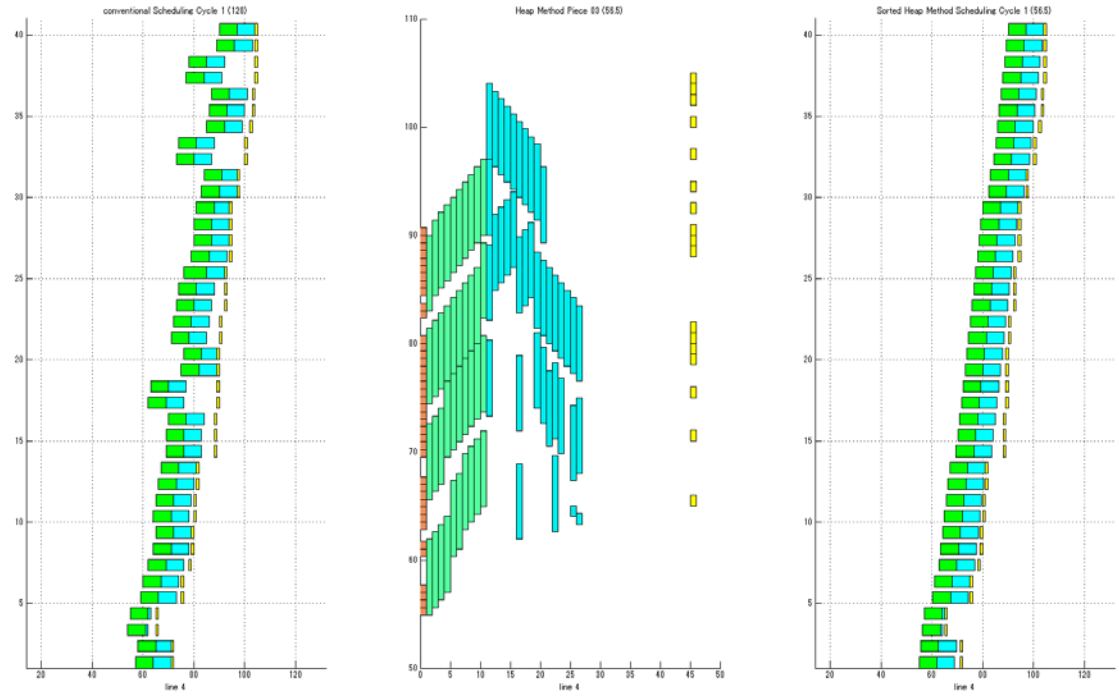

Fig.3-8 Cycle1 Line4 の現行計画(左)、Heap 図(中央)、適用後計画(右)

サイクル 1 の Line 5 のスケジューリングについて、現行計画のガントチャート、Heap 法適用後の Heap モデル図、ガントチャートを Fig.3-9 に示す。現行計画のガントチャートをみると、全体的にストック期間終了後から納期までに空き時間が発生しており、合計で 129 日のアイドル期間となった。

コンベア作業の開始日をみると、優先規則順にはなっておらず、前倒し気味に作業を始めている。これらのブロックをピースに変換し、納期へ向けて引き上げた結果が中央の Heap 図になる。コンベア作業をあらわすリソースは、ほぼ連続稼働する計画となっている。2 箇所、空き時間が生まれている場所があるが、上側の空き時間は納期日に接しているためにこれ以上引き上げられないブロックがあるためである。下側の空き時間は、ストックヤードの区画選択の結果、ピースの引き上げが阻まれ空き時間が生まれている。

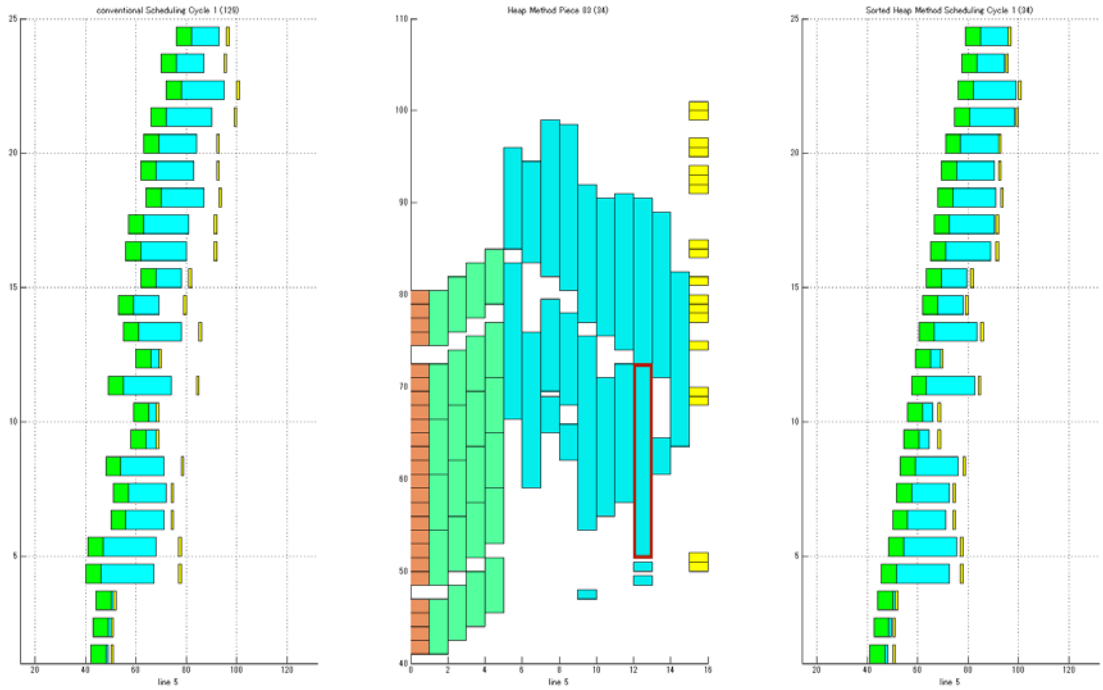

Fig.3-9 Cycle 1 Line 5 の現行計画(左) 、Heap 図(中央)、適用後計画(右)

ストックヤードを表すリソースは、割り当てたストック区画をすべて使い切る計画となっている。リソース選択規則 Rule 3 の結果、全体的にできるかぎり空き時間が生まれないように選択している。しかし一部ピースでは最良でない選択となっている。例えば、赤枠で囲ったピースは上にあるピースに阻まれ納期に接することができずにいる。もしこれが左隣のピースと逆に配置されていれば、納期に接するまで引き上げることができた。これは、ストックヤードの状況だけを考慮して左隣のピースが最良の区画を選択したが、コンベアが詰まっているためにこれ以上引き上げられず、それにより赤枠で囲ったピースにとって最良のリソースを塞ぐことになったためである。このため選択規則の改良の必要性があることが明らかとなった。

Heap モデル図をガントチャートにしたものが右の図となる。現行計画と比較すると隙間が詰まり、総アイドル期間は 35 日となっている。コンベアへの投入順序も優先規

則にそった順序になっている。サイクル 1 の Line 6 のスケジューリングについて、現行計画のガントチャート、Heap 法適用後の Heap モデル図、ガントチャートを Fig.3-10 に示す。現行計画のガントチャートをみると、一部ブロックを早めに組み立て始める計画となっており、アイドル期間は 286 日であった。これに対して Heap 法を適用した結果、アイドル期間が 402.6 日に増加した。

Heap 図を見るとストックヤードには空き時間が多くあるが、コンベアラインは連続稼働しているためにこれ以上詰めることができないことがわかる。適用後のガントチャートをみると優先規則に従って連続的にブロックが投入された計画になっていることがわかる。アイドル期間が増えた理由として、ブロックの同時投入の未考慮が考えられる。現行計画をみるといくつかのブロックがペアとなり同時にコンベア作業が始まっている。これは、定盤上でブロックを組み立てる際にブロックサイズが小さいものを並べ、定盤面積を有効活用することを目的としている。本検討では、この 1 定盤に 2 ブロックを想定せずに、1 定盤 1 ブロックとしてピースを作成し計画を行っている。これにより本来よりもアイドル期間を多く発生する計画となっていた。ピースを定義する際には、ブロック組み立てに関する工程ごとの特性を調査しピースに反映させることが必要だと明らかとなった。

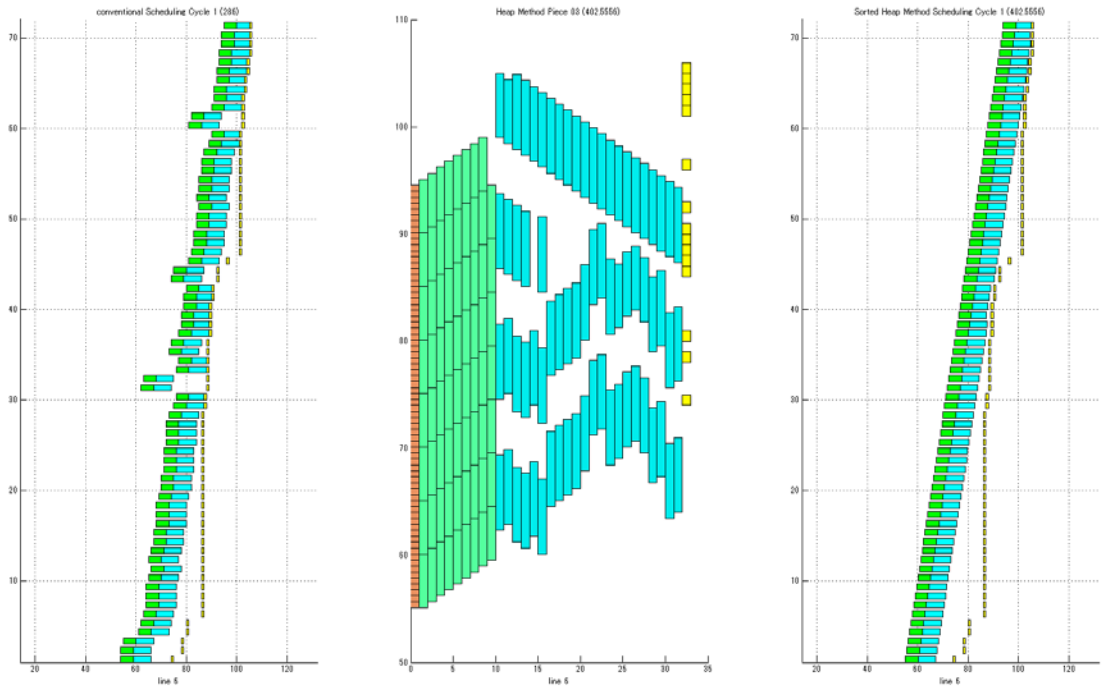

Fig.3-10 Cycle 1 Line 6 の現行計画(左) 、Heap 図(中央)、適用後計画(右)

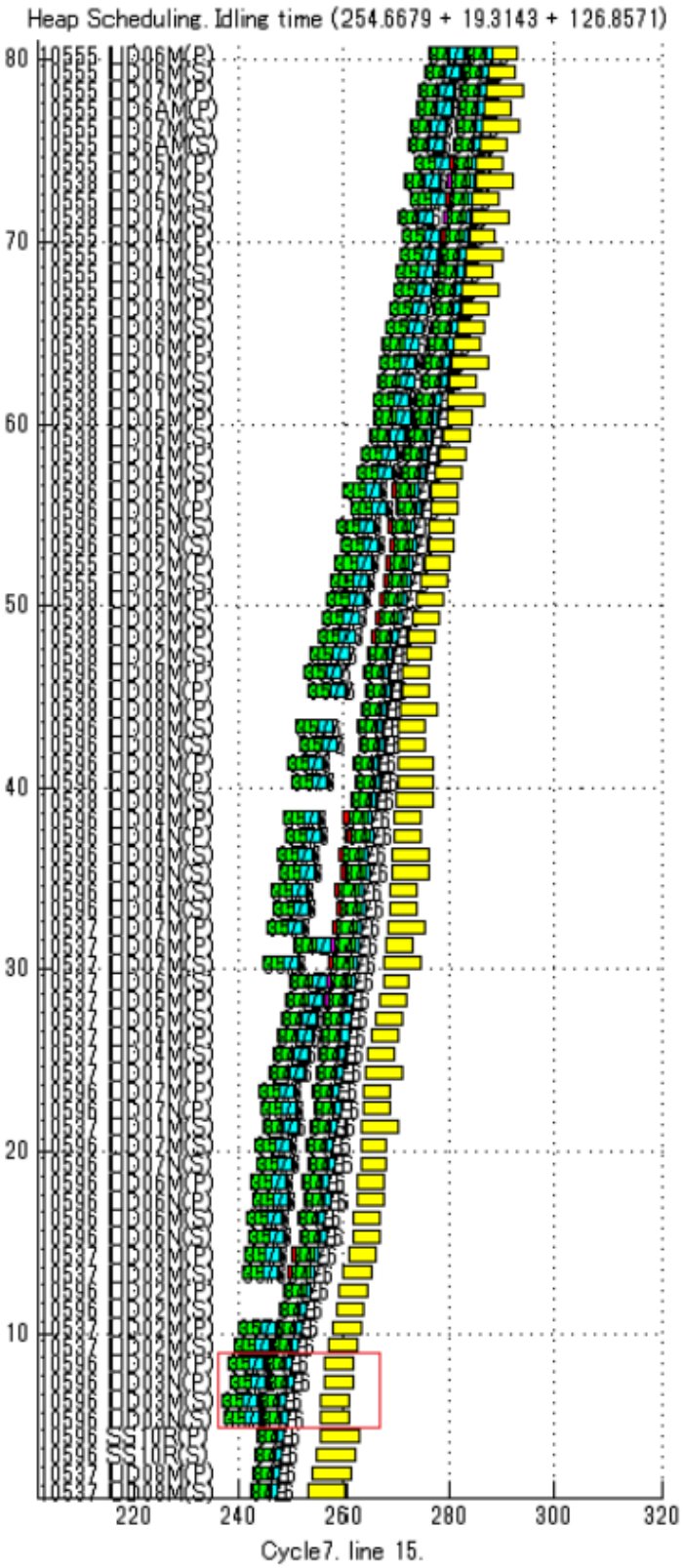

Fig.3-11 Cycle 7 Line15 全ブロックおよび Line 13 の先行中組ブロック

次に中組ブロック組立ラインの結果について述べる。中組ブロック組立工程では、大組工程の結果から製作の目標日が変化したため、一概にアイドル期間の変化を比較することができない。そのためここでは Heap 法での同期制約が実問題でも満足できているかという点に注目する。

当該造船所の中組ブロック組立ラインでは、Line 13 と Line 15 に先行中組ブロックが存在する。両ラインで組み立てられた先行中組ブロックは、一旦ストックヤードに運ばれたのち、然るべき時に Line 15 へ再投入され通常中組ブロック組み立ての基礎となる。そのため、今回のスケジューリングでは、先に Line 15 で作製される先行・通常中組ブロックのスケジューリングを行う。これにより求まった組立開始日を Line 13 の先行中組の目標日として、Line 13 の全てのブロックのスケジューリングを行う。この順序でプル型スケジューリングを行った工程計画結果のうち、サイクル 7 のものを Fig.3-11 に示す。これは Line 15 で制作される通常中組ブロックのガントチャートに、Line 13 と Line 15 で制作される先行中組ブロックのガントチャートを重ねたものである。先行中組ブロックを必要とする通常中組ブロックの場合、ガントチャートは左から先行中組ブロクのコンベア作業、ストック期間、通常中組ブロックのコンベア作業、ストック期間、納期の順に並んでいる。このガントチャートのうち、Line 13 から Line 15 へ送られるブロックの例として、四角の枠で囲った部分を拡大したものを Fig.3-12 に示す。

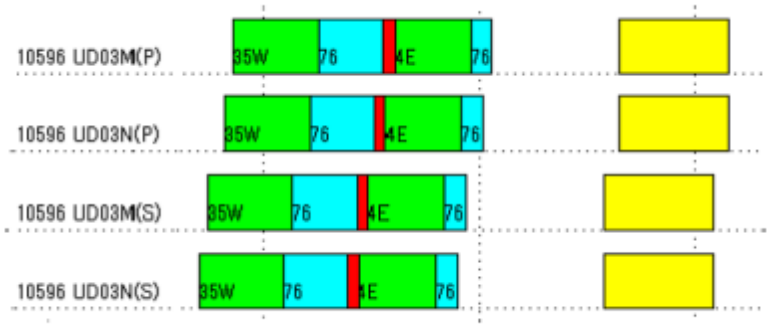

Fig.3-12 ライン間同期を考慮するブロックのガントチャート拡大図

Heap 法ではピースを順次引き上げて計画を行う、逐次処理を用いている。そのため、ピースを引き上げる前に目標日を確認し、必要であれば引き上げる目標となる下側境界線を更新することで、同期をとる。この処理によって、Line 15 で作製される通常中組ブロックのスケジューリングが完了することで開始日が確定する。この開始日を Fig.3-12 では赤のバーで示している。Line 13 のスケジューリング時にこの赤のバーで表した日に目標日を更新してピースを引き上げることができているため、先行中組ブロ

ックのストック期間が完了してから、通常中組ブロックのコンベア作業が開始する計画を作成できている。これは他の先行中組ブロックや、他のサイクルでも同様に確認することができた。これにより、Heap 法における同期制約の厳守を、実問題データでも確認することができた。

3.4 人員平準化を考慮した投入順序の検討

ここまでの検討では、納期にジャストインな計画を作成することで、アイドル期間を削減し工程のリードタイムを短縮することを目的としていた。しかしこの検討で得られた工程計画を実際の生産工程で実行することを考える時、造船所の作業員の配員を考える必要がある。作業量が多いブロックがある日にかたまると、造船所の作業員数を超えてしまい、作業が滞ることになる。作業が実行できるように多くの人員を抱えると、作業量の少ないブロックが続いた時に、遊んでいる人員がでるため人件費の面で無駄が生じてしまう。本節では、アイドル期間を削減して建造の効率化をはかりつつ、日々の作業量を平準化した工程計画を得るために、生産工程へブロックを投入する順序に注目した[13]。

ブロックの数をNとした時、投入順序は$N!$通りの組み合わせを持つ、組合せ最適化問題と考えられる。そこで最適化アルゴリズムである焼きなまし法を適用し、アイドル期間の最小化と人員平準化を達成する製作順序を探索する。

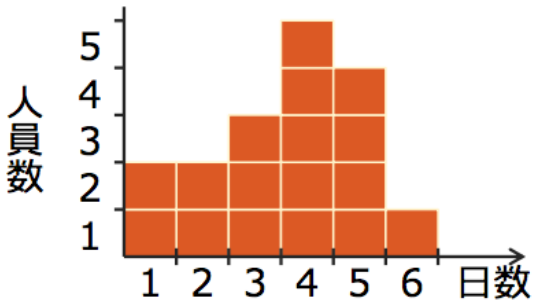

Fig.3-13 人員山積み表

この最適化問題の一番目の目的関数は、これまで扱ってきたアイドル期間である。あらためてアイドル期間を定義すると、ブロックiの納期をd_i、ストックヤード作業完了日をy_iとしたとき、

$$F_{idling\ time} = \sum_{i=1}^{N} (d_i - y_i) \tag{4.1}$$

と書ける。この目的関数$F_{idling\ time}$を最小化する必要がある。

本節では加えて、日々の人員数を平準化する必要がある。工程計画を立てた結果得ら

れるガントチャートに対し、各作業に必要な人員について作業日ごとに集計しグラフにしたものを人員山積み表と呼ぶ。この例を Fig.3-13 に示す。この山積み表を作成することで、作業人員の最大数や人員の偏りが明確になる。この図では第 4 日目では 5 人の作業者を必要としている反面、第 6 日目で必要な作業者は 1 人である。通常、ピーク時の人員数を常時確保しておく必要があるが、ここで作業人員の稼働率を考える。文献に従って稼働率rを

$$r = \frac{h_{all}}{h_{max} \times t} \tag{4.2}$$

と定義する。ここでh_{all}は作業を行うのに必要な総人員数、h_{max}はピーク時の人員数、tは作業日数を意味する。Fig.3-13 に示す山積み表について、この稼働率を計算すると

$$\begin{cases} h_{all} = 17 \\ h_{max} = 5 \\ t = 6 \end{cases} \tag{4.3}$$

より

$$r = \frac{17}{5 \times 6} = \frac{17}{30} = 0.567 \tag{4.4}$$

となる。つまりこの計画では、56.7%の人員しか作業に関わっていないということになる。

ここで、工程計画に修正を加えて人員平準化を行った結果、Fig.3-14 のような山積み表が得られたと仮定する。全体の作業日数と、総人員数は変更がなく、ピーク時の人員が 5 人から 4 人となっている。稼働率を計算すると

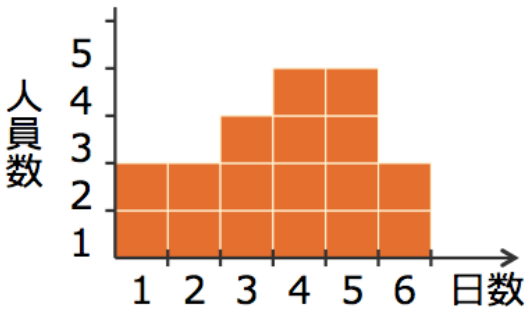

Fig.3-14 人員平準化後の山積み表

$$\begin{cases} h_{all} = 17 \\ h_{max} = 4 \\ t = 6 \end{cases} \tag{4.5}$$

より

$$r = \frac{17}{4 \times 6} = \frac{17}{24} = 0.708 \tag{4.6}$$

となる。平準化の結果、稼働率が 56.7%から 70.8%へ向上した。このように稼働率の向上には、作業日数の短縮とピーク時人員の最小化が効果的である。本研究ではこのピーク時人員数を二番目の目的関数とし

$$F_{peak} = h_{max} \tag{4.7}$$

と定義する。

このようにこのスケジューリング問題は、多目的組み合わせ最適化問題となる。本研究では、各目的関数に対して重みwを用いて、単目的関数へと落としこむことを行った。ただし重みの合計は 1 となることに注意する。

$$F = w_1 F_{idling\ time} + w_2 F_{peak} \tag{4.8}$$

この重みFを最小化するような工程計画を得るために、ブロックの投入順序を焼きなまし法を用いて調査する。

本節では、ブロック組立計画データのうち、造船所から作業時数に関するデータが提供された Line 3 について取り組んだ。ここからは作業人員に作業時間を掛けあわせた作業時数を基に稼働率を計算していく。本検討では、サイクル 1 からサイクル 9 のそれぞれに、目的関数の重みを変更しつつ焼きなまし法プログラムを適用した。ひとつのサイクルにおいて、重みの条件を 6 パターン用意し、各条件につき 10 回ずつ、合計 60 回適用した。このときの重みについて Table 3-5 についてまとめる。

Table 3-5 目的関数の重み係数

	No.1	No.2	No.3	No.4	No.5	No.6
w_1	0.99	0.90	0.80	0.70	0.60	0.50
w_2	0.01	0.10	0.20	0.30	0.40	0.50

焼きなまし法を適用した結果のうち、サイクル 5 の 1 回目の計算仮定を Fig.3-15 に示す。上段のグラフはアイドル期間の変化を示している。計算回数が進むごとにアイドル期間は小さくなっている。重み条件 No.1 を意味する青線を見ると、現行計画のアイドル期間より小さいアイドル期間を求めることができている。重み条件をアイドル期間から人員平準化に傾けていくにつれ、アイドル期間の値が大きくなっている。

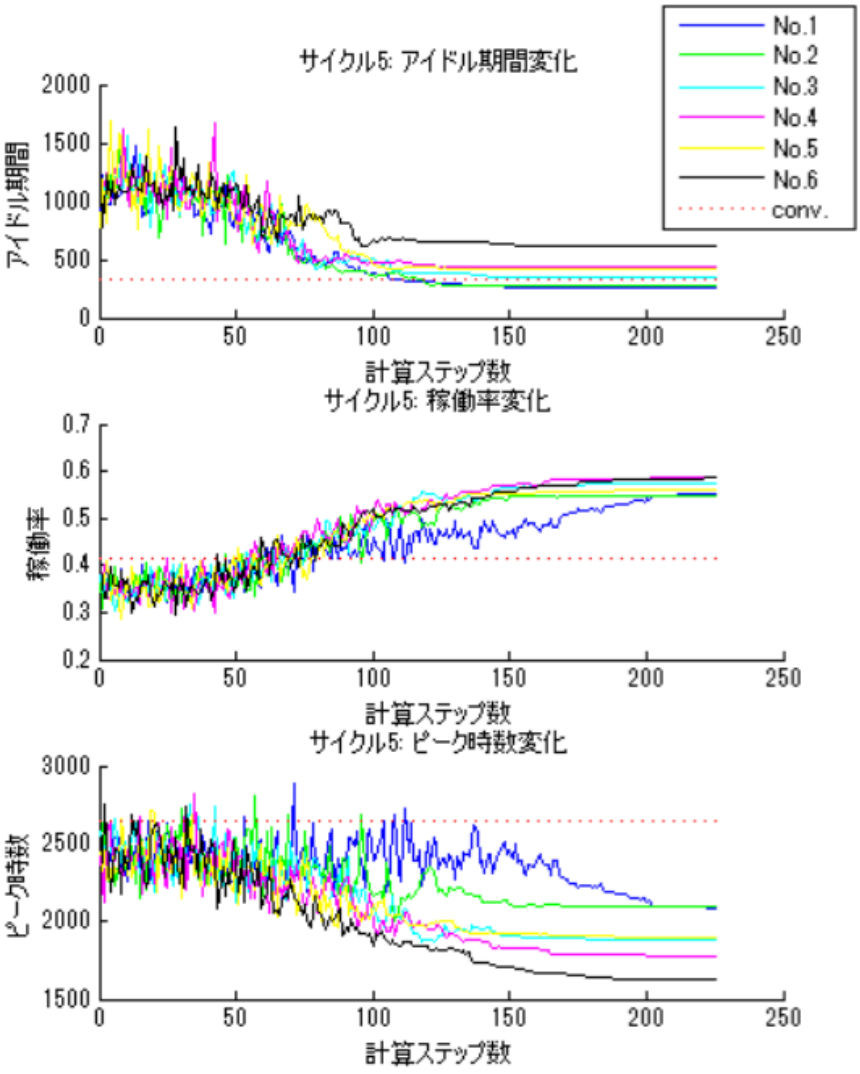

Fig.3-15 Cycle 5 の場合の焼きなまし法の計算経過

中段および下段のグラフは稼働率の変化とピーク時数の変化を示している。計算回数が進むにつれ、ピーク時人員数が減少していき、それに応じて稼働率が上昇している。アイドル期間に重みを大きくかけている No.1 ではピーク時数が 2100 時数付近で収束している。人員平準化に傾けるにつれ、ピーク時時数が下がる様子が見て取れる。このようにアイドル期間を抑えつつ人員平準化を行う意図にそった結果が得られている。

次に、Fig.3-16 に、各サイクルにおいて各重み条件で 10 回計算した際の最終結果の値をプロットしたグラフを示す。グラフの縦軸に稼働率、横軸にアイドル期間をとる。各点の色は Fig.3-15 と同様になっている。

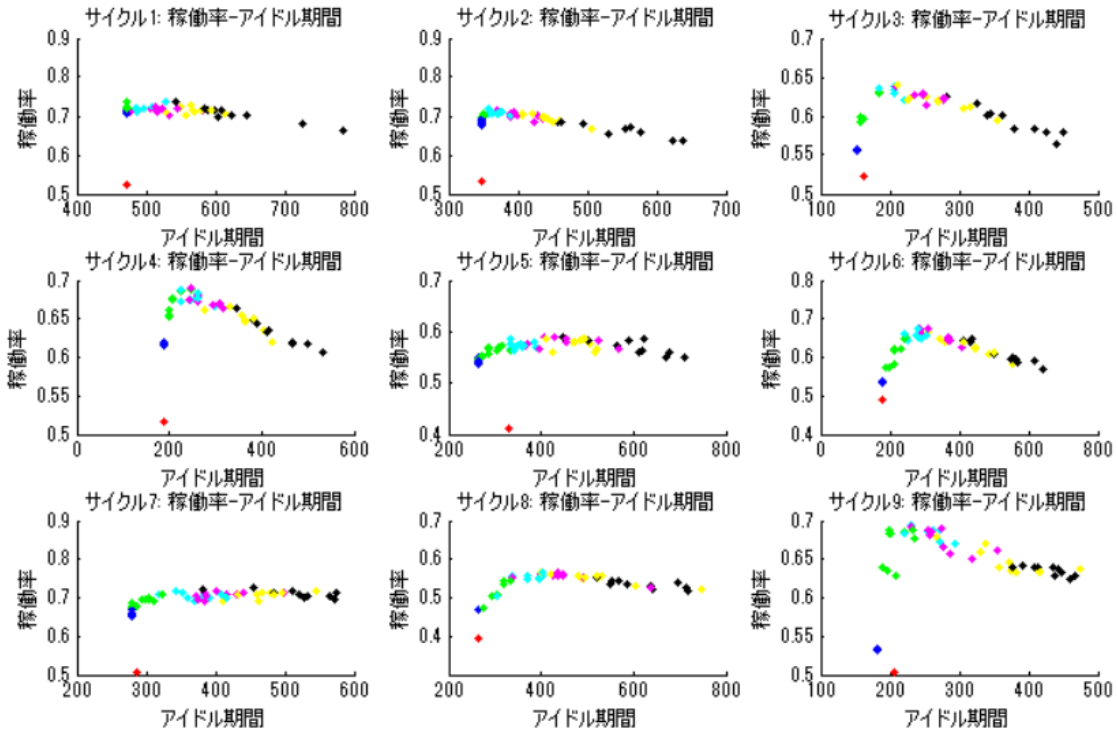

Fig.3-16 各サイクルの SA 適用結果

すべてのサイクルで、現行の投入順序決定規則である優先規則を上回る稼働率と、アイドル期間の削減を達成できる投入順序があることがわかった。サイクル 1 と 7 を除いて、アイドル期間を削減するような投入順序を採用すると稼働率が下がり、アイドル期間を許容するにつれ稼働率が上昇、さらにアイドル期間を許容しすぎると稼働率が悪化する、上に凸の傾向を示している。

サイクル 5 について得られた結果のうち、優先規則に基づき投入した結果を Fig.3-17 に、アイドル期間最小となった投入順序の結果を Fig.3-18 に、稼働率最大となった投入順序を Fig.3-19 に示す。それぞれ左側にガントチャート、右側に各ブロックの時数を示している。ガントチャートは上から順に、ピースを引き上げた順序となっており、ブロックの組立順序は下から順に見ていけば良い。

優先規則に基づき投入した結果、アイドル期間は 330.7 日、稼働率は 0.41 となった。Fig.3-17 のガントチャートと対応するブロックの時数を見ると、投入順序の終盤に時数の大きくかかるブロックが固まっている。また同じ時数のブロックが隣り合って並んでいる。その結果、ピーク時数が大きくなり稼働率が低下した。

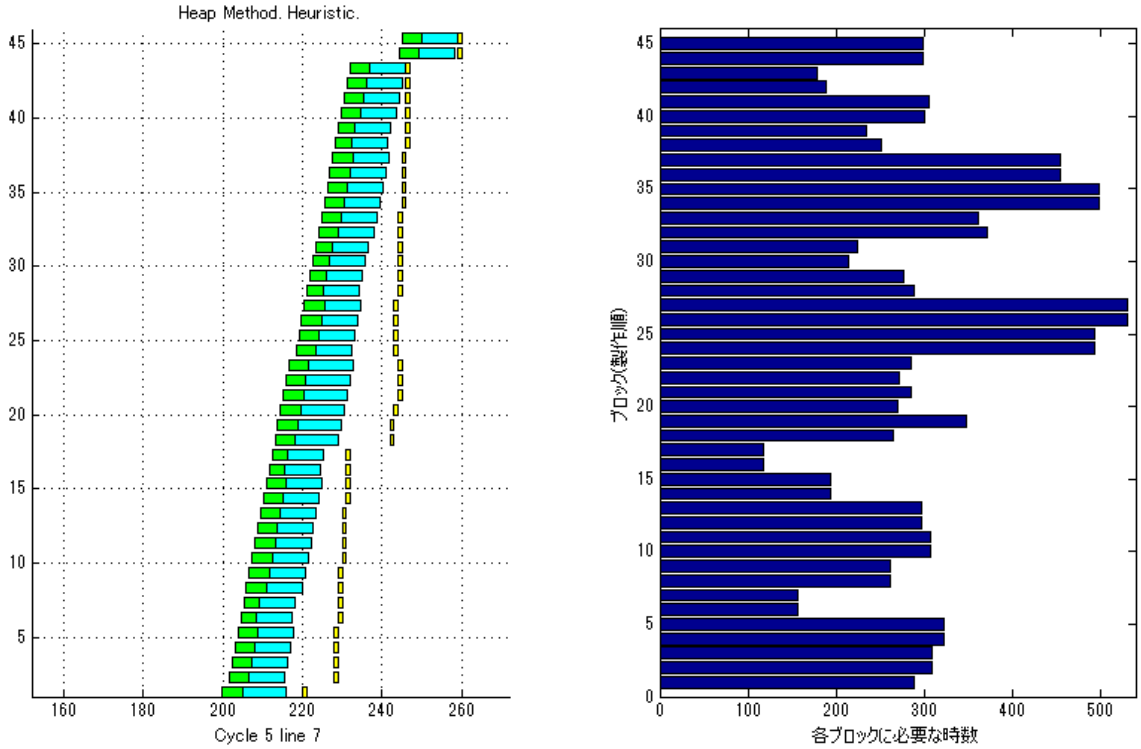

Fig.3-17 優先規則による投入順序を用いた計画結果

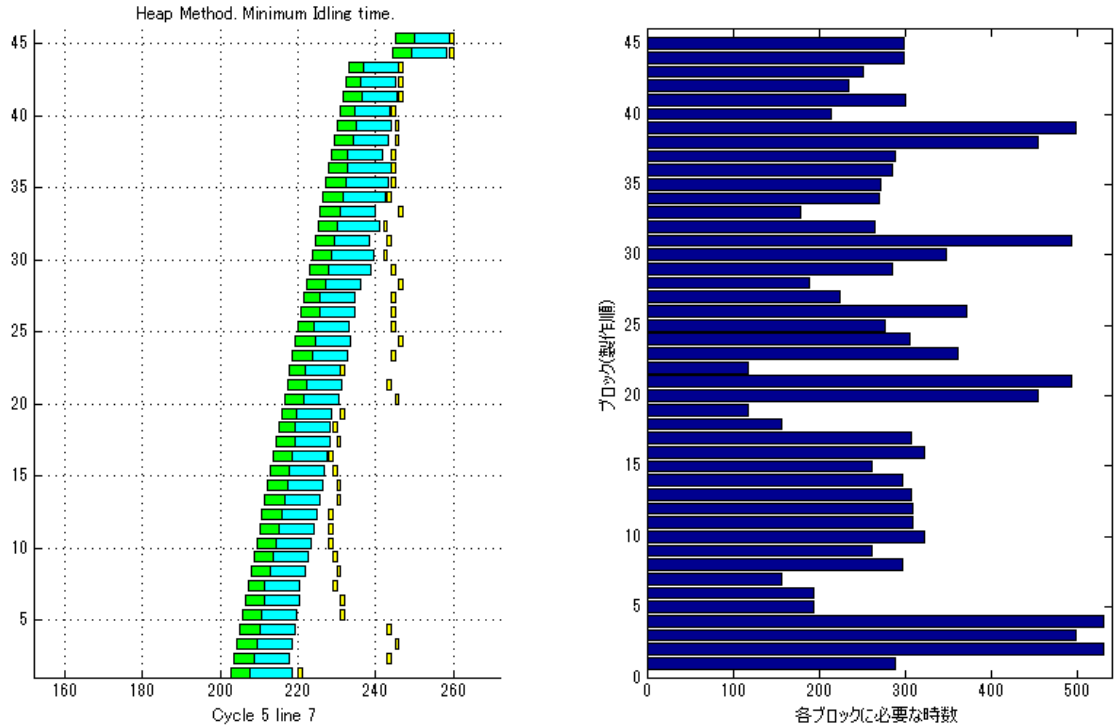

Fig.3-18 アイドル期間最小となる投入順序を用いた計画結果

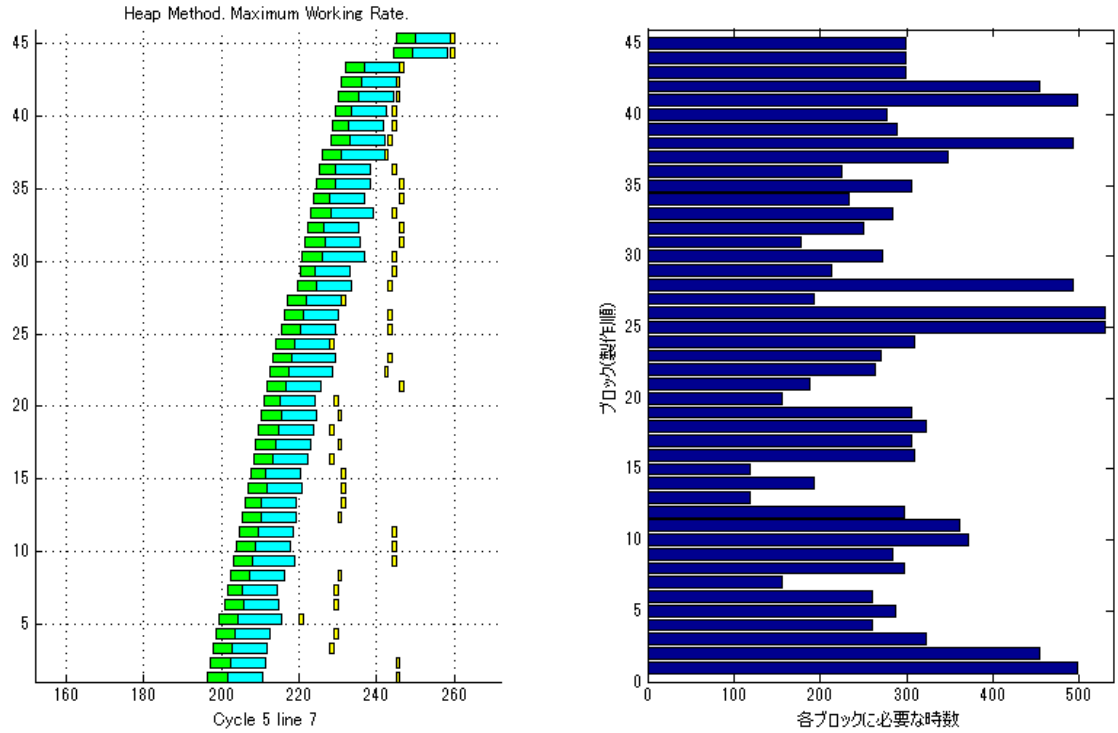

Fig.3-19 稼働率最大となる投入順序を用いた計画結果

アイドル期間を最小にした投入順序を採用した結果、アイドル期間は 264.0 日、稼働率は 0.55 となった。優先規則と比較するとアイドル期間が 66.7 日の削減、稼働率が 0.14 の上昇となった。Fig.3-18 のガントチャートとブロック時数を見ると、納期の遅いブロックをいくつか先に製作していることがわかる。一部のブロックのアイドル期間を増加させることで、他のアイドル期間をなくすことができ、その結果全体のアイドル期間が減少したと考えられる。また各ブロックの時数に注目すると、前倒しされたブロックの時数が優先規則では終盤に固まっていた時数の大きいブロックであったために、時数のかかるブロックが全体に分散した。その結果、ピーク時数が小さくなり稼働率が向上した。

稼働率を最大にする投入順序を採用した結果、アイドル期間は 448.2 日、稼働率は 0.59 となった。優先規則と比較するとアイドル期間が 117.5 日の増加、稼働率が 0.18 の上昇となった。Fig.3-19 のガントチャートとブロック時数を見ると、Fig.3-18 の場合よりも、さらに多くの納期の遅いブロックを前倒しした投入順序となっている。この結果、時数のかかるブロックを更に分散させ、ピーク時数を下げようとしている。このような時数のかかるブロックの分散は、工数管理において重要な操作となっているが、最適化手法でもそのような解を好むことが明確となった。反面、納期の遅いブロックを早い段階で多く作りすぎたために、アイドル期間を大きく増やす結果をとなっている。

今回の検討では、アイドル期間と人員平準化を考慮した工程計画を行うため、多目的

組合せ最適化問題として投入順序の調査を行った。その結果経験則である優先規則よりも少ないアイドル期間を得つつ、人員平準化できる投入順序があることがわかった。アイドル期間と人員稼働率の関係性については、事前の想定通り相関関係があり一方を向上させようとすると、もう一方が犠牲になることが明確となった。アイドル期間最小によるリードタイムの短縮を必要とするのか、工程内の作業人員の稼働率向上を必要とするのかは、組立工程計画を立てる造船所の状況に依るため、今回のようなパレート解を得て、そこから適切な解を選択する必要がある。

3.5 考察

本章では、造船所 A の実問題データに対して、Heap 法によるプル型スケジューリングを適用しその効果を検証した。まず現行の計画と同様に優先規則に従った投入順序で Heap 法を適用した。その結果、設備制約および同期制約を遵守した状態で、アイドル期間を大きく削減することができた。次にアイドル期間の削減だけでなく作業人員の平準化を考慮した工程計画を目指し、投入順序最適化を行った。これにより、アイドル期間と稼働率を目的関数としたパレート解を得て、アイドル期間を抑えつつ平準化された工程計画を得ることができた。

第 4 章 造船所 B の組立工程計画検討

4.1 造船所 B の概要

Fig.4-1 に造船所 B のレイアウトを示す。また、各作業区画で行われる作業内容について、Table 4-1 に示す。

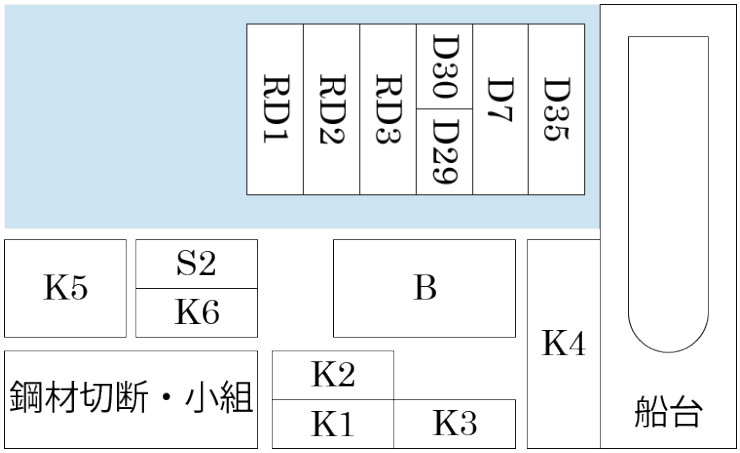

Fig.4-1 造船所 B の組立工程レイアウト

Table 4-1 各作業区画での作業内容

作業区画	作業内容
K1, K2	取付、溶接
K3, K4	組立、溶接、先行艤装、先行塗装
K5, K6	取付、溶接
B1 ~ B3	ブラスト、塗装
S2, D7~35, RD1~ RD3	待機、先行艤装、先行塗装

K1、K2 および K5 は建屋内に固定定盤を持つ作業区画で取付と溶接を行う。S2 および K3、K4、K6 は屋外に固定定盤を持つ作業区画で、複数ブロックを組み合わる、ブロックの反転正転、先行艤装を行うといった作業を行う。B1 ~ 3 は塗装工場で、ブラスト処理と塗装処理を行う。D7 ~ 35 や RD1~3 は海上に浮かべられたバージ(台船)で、基本的に塗装や搭載までの間保管するための場所である。バージの外観を Fig.4-2 に示す。搭載待ちのブロックが多くなった場合や外注したブロックの入荷日が重なった場合にバージに保管できる量を超えた場合は、バージを借りて追加する事ができる。

Fig.4-2 海上バージ

この造船所内でのブロック製作の流れを、各作業場の詳細を説明しつつ確認する。造船所へ搬入された鋼板は、切り出し場にて部材の切り出しが行われる。次に切り出された部材は K1、K2 もしくは K5 へ搬入される。主に船殻に関わるブロックの組立が K1、K2 から始まる組立工程で行われ、タンクやコルゲートバルクヘッドといった船体内部に関わるブロックの組立が K5 から始まる組立工程で行われる。

Fig.4-3 建屋内作業場 K1 の様子

まず、K1 および K2 から始まる組立工程について述べる。Fig.4-3 に、建屋内作業区画 K1 および K2 の写真を載せる。ここではそれぞれ 4 区画、計 8 区画の固定定盤が配備されている。切り出し場から搬入された部材を鉄板に取り付け、溶接し小さいブロックを作成する。ここは建屋内固定定盤であるため、完成したブロックは天井クレーンで持ち上げ搬出される。この時、他のブロックの上を越えていく必要があるため、定盤上のブロックとクレーンに釣られたブロックの高さの合計が 7 メートル以内におさまる

ように定盤計画を行う必要がある。ここでの作業が終わるとブロックはK3へ送られる。

Fig.4-4 屋外作業区画 K3 の様子

Fig.4-5 屋外作業区画 K4 の様子

Fig.4-4 に屋外作業区画 K3、Fig.4-5 に屋外作業区画 K4 の写真を載せる。K3、K4 共に作業定盤が 5 区画配備されている。前工程から送られてきたブロックの正転・反転のために大型のクレーンが設置されており、2 つのブロックを組み合わせ溶接する作業を主に行う。次のブロックまで時間がある場合は、この時点で先行艤装、先行塗装、社内検査、船級検査といった作業を行う。次の作業が詰まっておりすぐに動かさなければならない場合は、バージへ送られて、そこで検査等の作業を行う。ここでの作業が終わ

ると塗装工場へと送られる。もし塗装工場に空きがなければバージに送られ蔵置される。

Fig.4-6 建屋内作業区画 K5 の様子

Fig.4-7 屋外作業区画 K6 の様子

次に K5 から始まる組立工程について述べる。Fig.4-6 に建屋内作業区画 K5 の写真を載せる。ここには固定定盤が 6 区画配備されている。ここでは切断された部材を鋼板に取付・溶接作業を行う。完成したブロックは天井クレーンで K6 もしくは S2 へ搬出される。

Fig.4-7 に屋外作業区画 K6 の写真を載せる。ここには固定定盤が 4 区画配備されている。ここでは前工程から搬入されてきたブロックを組み合わせてさらに大きなブロックへと組み上げる。ここで完成したブロックは台車へ載せられ塗装工場へ運ばれるか、塗装の必要がないものはバージ上で搭載日を待つことになる。

最後に造船所内のブロック物流の概要を Fig.4-8 に示す。これまで述べてきた K1、K2 から始まる船殻関連ブロックの流れと、K5 から始まる船内関連ブロックの流れに加え、外注したブロックの流れを表記している。外注したブロックは、ある工場に発注した複数ブロックがまとめて洋上を運搬される。造船所内に入荷するとバージ上に保管され艤装品が取り付けられる。その後塗装が必要なブロックは塗装工場へ送られ、搭載日が来ると船台へ搭載される。

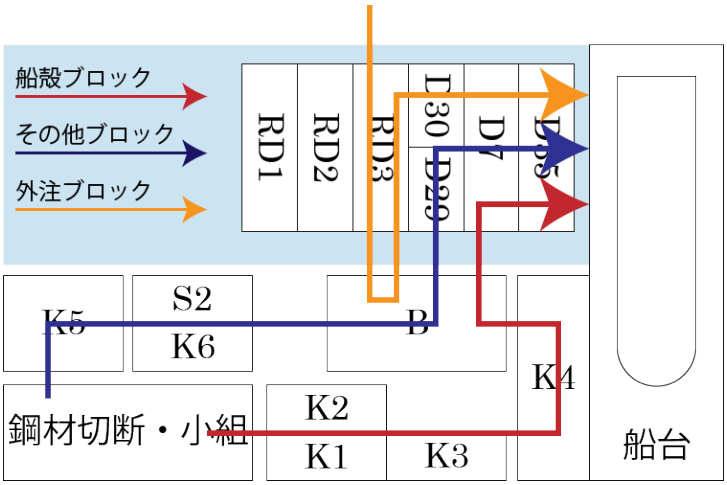

Fig.4-8 造船所内ブロック物流

以上のような造船所内のブロック物流を考慮して、この造船所のリソースとブロック組立のピースを定義する。

4.2 組立工程の Heap モデル

この造船所 B に対して Heap 法によるプル型スケジューリングを適用するために、リソースとピースの定義を行う。

当該造船所のうち、今回の研究で取り扱う工程を Table 4-2 に示す。Fig.7-001 に示した造船所レイアウトの概略図内の切り出し場と船台を除く全ての工程を扱う。4 章で扱った造船所 A では、各組立工程を個別のラインとみなしてモデル化を行った。しかし、造船所 B は組立ラインのあいだにバッファがなくひと繋がりのラインとみなせる。そのため Table 4-2 に示した工程をリソースとして並べ、造船所全体をひとつのモデルに落としこむ。

Table 4-2 対象とする工程と区画数

工程名	区画数	工程名	区画数
K1	4	RD1	4
K2	4	RD2	4
K3	5	RD3	5
K4	5	D7	8
K5	6	D29	3
K6	4	D30	4
B	3	D35	8
S2	3	F	2

この造船所でのブロック組立工程のピースを定義する。ブロック組立工程の流れは前節で述べたように 3 種類ある。これらの流れを、K1 または K2 から始まる船殻ルート、K5 から始まる船内ルート、外注ブロックの入荷から始まる外注ルートとここでは呼ぶ。

最初に船殻ルートのピース定義を行う。船殻ルートでの作業順序は主に、取付・溶接・正反転・溶接・先行艤装・社内検査・船級検査・塗装の順で行われる。ブロックの種類によって取付・溶接作業が追加されたり、一部作業が削除されたりする。船殻ルートは 2〜4 つのピースに分けて考える。これは、このルートでの工程の連続性とバッファの位置を考慮すると、K 系工程上の取付から検査作業、D もしくは RD 系工程上の艤装作業、B 系工程上のブラスト塗装作業、D もしくは RD 系工程上の塗装後艤装作業の 4 つの作業集団に分けることができるためである。

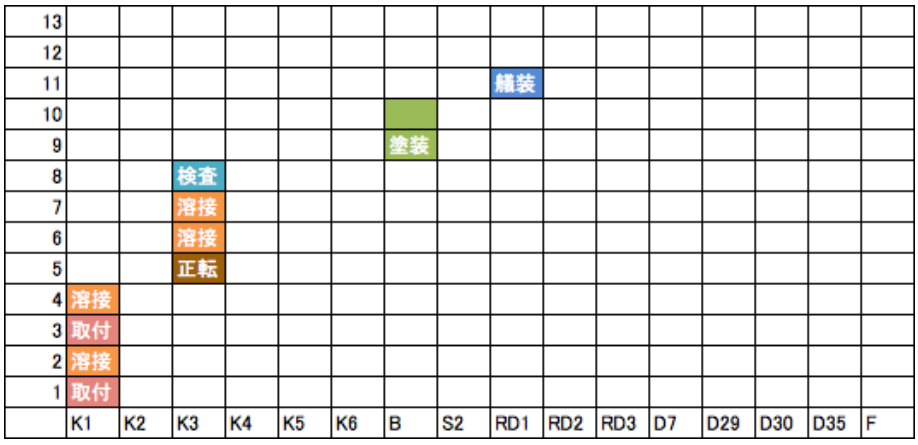

Fig.4-9 船殻ルートを通過するブロックのピース

ここでは例として、3 つの作業集団で構成されるブロックを 3 つのピースとして定義し、Fig.4-9 に示す。Fig.4-9 は K1 と K3 での作業でひとつのピース、B での作業でひ

とつのピース、RD1 での作業でひとつのピースと、合計 3 つのピースを並べて表示したものである。K3 および K4 と塗装工場 B への間に待機時間が発生しても海上のバージに蔵置ができるため塗装の直前で一旦区切っている。また塗装後もバージに蔵置されるため、同様に区切っている。そのためスケジューリング後には Fig.4-10 のように間にアイドル期間が発生する場合がある。

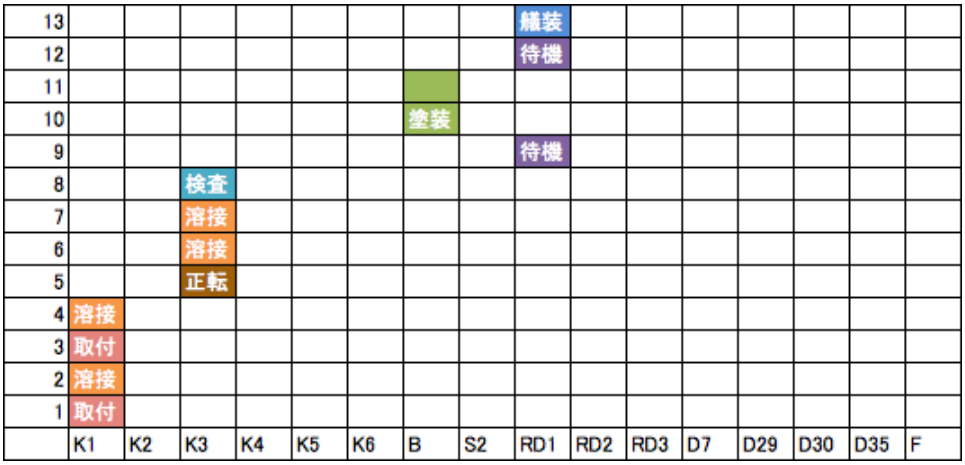

Fig.4-10 スケジューリングによりアイドル期間の発生する例

次に船内ルートのピース定義を行う。このルートでの作業順序は船殻ルートと同様に取付・溶接・正反転・溶接・先行艤装・社内検査・船級検査・塗装の順で行われる。船内ルートのピースも船殻ルートと同様に考えることができるため、塗装前の作業で 1 ピース、塗装で 1 ピース、塗装後で 1 ピースの合計 3 つのピースでほぼ定義できる。船内ルートの 3 つのピースをまとめて表示したものを Fig.4-11 に示す。

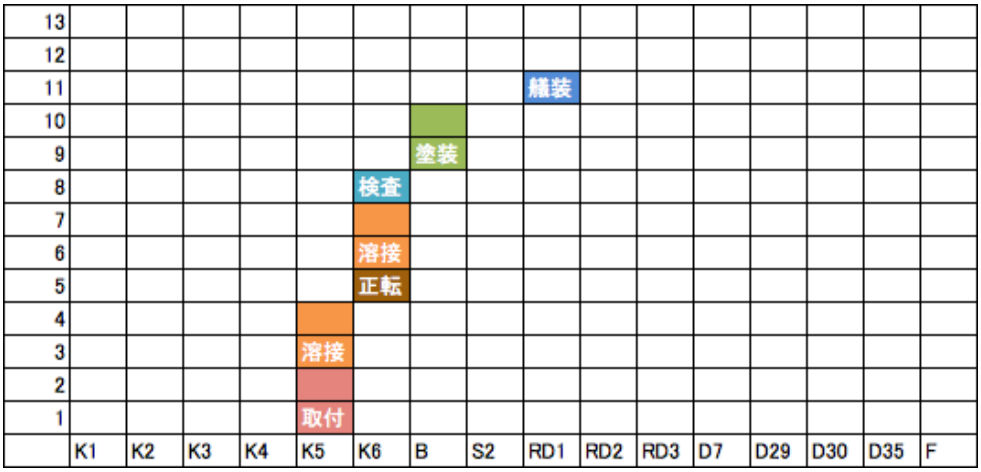

Fig.4-11 船内ルートを通過するブロックのピース

Fig.4-11 の例に示すように K5 から始まる船内ルートも、Fig.4-9 と同じように取付溶接から始まり、艤装で終わる流れになっている。こちらも各ピースを個別に引き上げる関係で、スケジューリングの結果によってはピースの境にアイドル期間が発生する場合

がある。この例を Fig.4-12 に示す。

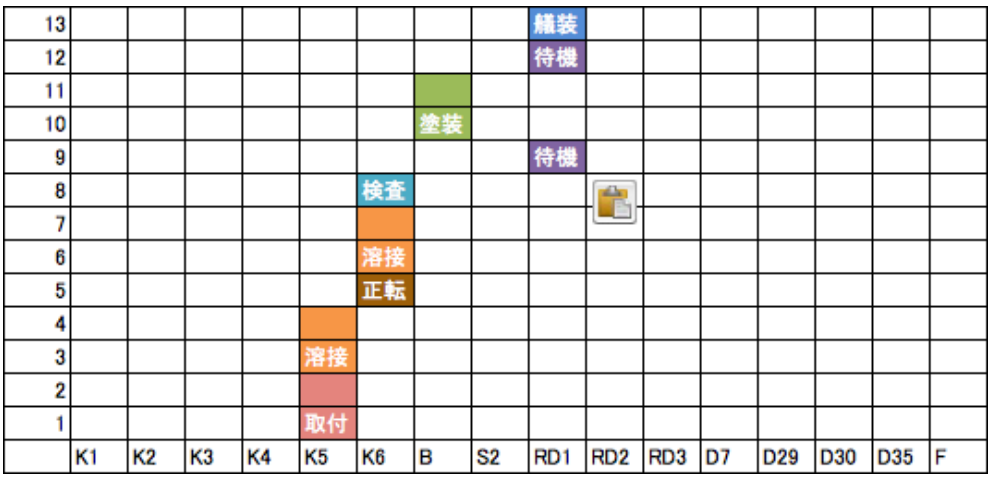

Fig.4-12 スケジューリングによりアイドル期間の発生する例

最後に外注ルートのピース定義を行う。外注ルートでは、入荷・塗装・艤装か、入荷艤装の作業手順をたどる。これは外注先で塗装まで行えるかどうか、もしくは塗装が必要なブロックかどうかで分かれる。そのためブロックの組立を表すピースは、塗装がない場合は 1 つ、塗装がある場合は 2 つのピースで構成される。ここでは塗装がある場合のピースの例を Fig.4-13 に示す。

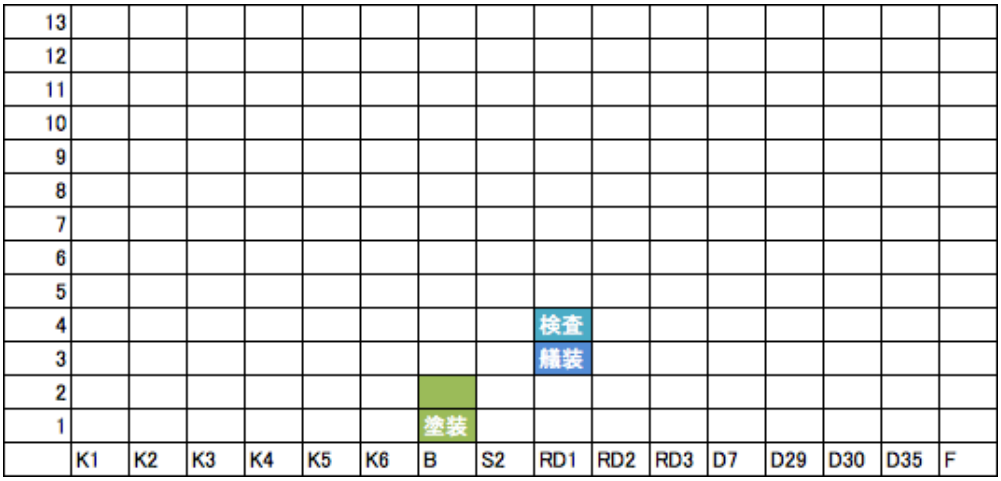

Fig.4-13 外注ルートを通過するブロックのピース

塗装がある場合は塗装作業のピースと艤装・検査のピースの 2 つで構成される。ピース上には入荷日を定義していない。これは運搬費の関係で、外注先で完成した複数のブロックを同時に運んでくるため、他のブロックと入荷日合わせる必要があるためである。そのため、スケジューリング結果によっては入荷日から塗装日までのあいだにアイドル期間が発生することに注意する必要がある。2 つのピースは個別に引き上げて計画するために、両者の間にもアイドル期間が発生する場合がある。この例を Fig.4-14 に示す。

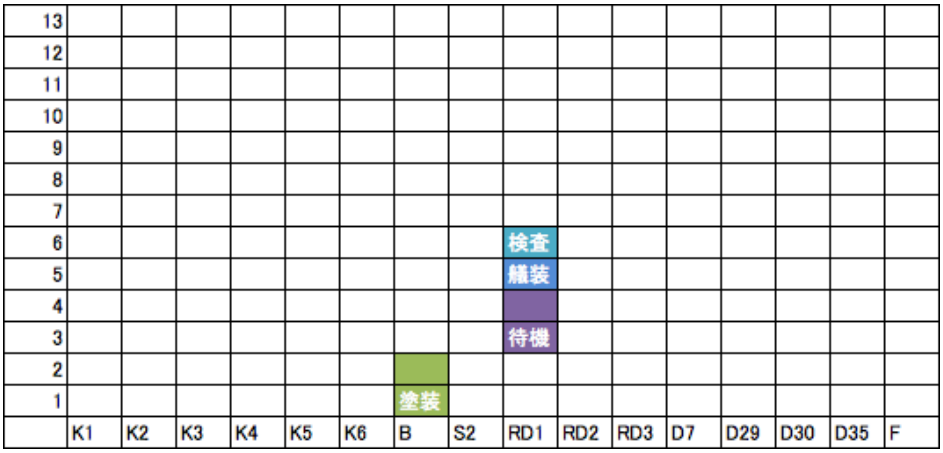

Fig.4-14 スケジューリングによりアイドル期間が発生する例

以上のように、この造船所 B では造船所全体をリソースとして定義して、各組立工程のルートにしたがってピースを定義してプル型スケジューリングを行う。実際にある番船のブロック組立計画の実データを基に、現行計画を Heap 図にて表現したものを Fig.4-15 に示す。この時各色は Table 4-3 に示す作業内容を示していることに注意する。この時利用した実データを Heap 法プログラム入力形式に整えたものの一部を Table 4-4 に示す。Heap 図の縦軸は日数を表している。横軸のリソースは、Fig.4-14 のように折りたたんでいたリソースを各区画数まで詳細に展開して表示している。

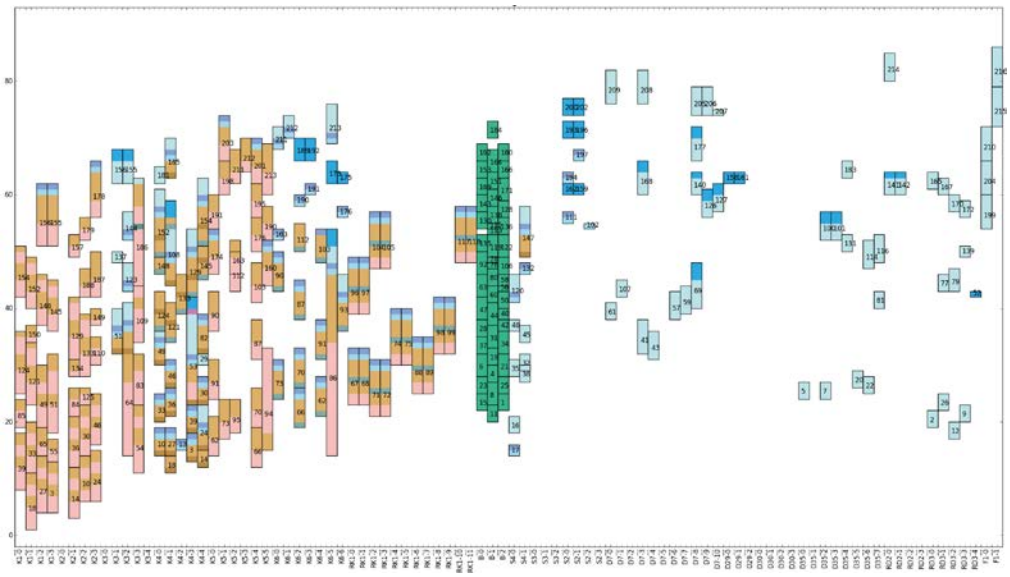

Fig.4-15 現行計画の Heap 図表現

各ピース内に表示している番号は、事前にピースに割り振った番号のことで、Table 4-4 に示す実データの No.の列にあたる。数字が同じ番号であれば同じピースの塊となっている。

Table 4-3 Heap 図の色と作業内容の対応表

色	作業内容	色	作業内容
	取り付け		溶接
	正転		反転
	社内検査		船級検査
	塗装・ブラスト		先行艤装
	先行塗装		待機

Table 4-4 ブロック組立データ表の例

No.	Name	Piece ID	Link	Due Date	Job ID	Job Name	Workload	Resource	Resource	Sub	Lower	Upper
1	5S1S	5S1S-1	0	41	8	ブラスト	0	1	8	3	22	25
2	5S1S	5S1S-2	1	41	6	艤装	0	1	17	1	19	22
3	5S1S	5S1S-3	2	41	0	取り付け	0	1	1	4	4	7
3	5S1S	5S1S-3	2	41	1	溶接	0	1	1	4	7	9
3	5S1S	5S1S-3	2	41	0	取り付け	0	1	1	4	9	11
3	5S1S	5S1S-3	2	41	1	溶接	0	1	1	4	11	13
3	5S1S	5S1S-3	2	41	2	正転	0	1	4	4	13	14
3	5S1S	5S1S-3	2	41	1	溶接	0	1	4	4	14	16
3	5S1S	5S1S-3	2	41	4	社内検査	0	1	4	4	16	17
3	5S1S	5S1S-3	2	41	5	船級検査	0	1	4	4	17	18
4	4S1P	4S1P-4	0	42	8	ブラスト	0	1	8	2	27	30
5	4S1P	4S1P-5	4	42	6	艤装	0	1	15	1	24	27
6	4S1S	4S1S-6	0	42	8	ブラスト	0	1	8	1	28	32
7	4S1S	4S1S-7	6	42	6	艤装	0	1	15	3	24	27
8	5S1P	5S1P-8	0	42	8	ブラスト	0	1	8	2	23	27
9	5S1P	5S1P-9	8	42	6	艤装	0	1	17	4	20	23
10	5S1P	5S1P-10	9	42	0	取り付け	0	1	2	3	6	8
10	5S1P	5S1P-10	9	42	1	溶接	0	1	2	3	8	10
10	5S1P	5S1P-10	9	42	0	取り付け	0	1	2	3	10	12
10	5S1P	5S1P-10	9	42	1	溶接	0	1	2	3	12	14
10	5S1P	5S1P-10	9	42	2	正転	0	1	4	1	14	15
10	5S1P	5S1P-10	9	42	1	溶接	0	1	4	1	15	17
10	5S1P	5S1P-10	9	42	4	社内検査	0	1	4	1	17	18
10	5S1P	5S1P-10	9	42	5	船級検査	0	1	4	1	18	19

現行計画を Heap 図で表してみると、塗装工場が隙間時間なく稼働しており、生産工程のボトルネックとなっていることがわかる。これは造船所の建設当時と現在の塗装基準に変更があり、これまで塗装の必要がなかったブロックも塗装作業が追加され塗装工場を通過することになったことが原因のひとつである。

この現行計画に対してプル型スケジューリングを適用してアイドル期間の削減とリードタイムの短縮を行っていくが、そのためにこの造船所用に設備制約を追加する必要がある。前節で説明したとおり、この造船所の組立建屋 K1 および K2 は一般的な造船所に比べて高さが低い。そのため、天井クレーンを用いて完成したブロックを搬出する場合、釣り上げたブロックと定盤上にある他ブロックの高さの合計が 7 メートル未満になるように定盤計画を行う必要がある。そのため、ブロックの高さ管理を行う処理を追

加した。

船殻ルートを通過するブロックについて、造船所より Table 4-5 に示すようなブロックサイズデータを得た。左の列から順にデータ番号、ブロック名、ピース識別 ID、ブロックの x・y・z 方向の大きさ、定盤上での高さ、重さとなっている。

Table 4-5 ブロックサイズデータ表の例

No.	Name	Piece ID	x	y	z	Height	Weight
1	5S1S	5S1S-1	12.811	13.35	1.89	1.89	130
2	5S1S	5S1S-2	12.811	13.35	1.89	1.89	130
3	5S1S	5S1S-3	12.811	13.35	1.89	1.89	130
4	4S1P	4S1P-4	12.058	10.35	1.89	1.89	126
5	4S1P	4S1P-5	12.058	10.35	1.89	1.89	126
6	4S1S	4S1S-6	12.115	13.351	1.89	1.89	126
7	4S1S	4S1S-7	12.115	13.351	1.89	1.89	126
8	5S1P	5S1P-8	12.65	10.68	1.89	1.89	130
9	5S1P	5S1P-9	12.65	10.68	1.89	1.89	130
10	5S1P	5S1P-10	12.65	10.68	1.89	1.89	130
11	6S1P	6S1P-11	12.7	10.68	1.89	1.89	130
12	6S1P	6S1P-12	12.7	10.68	1.89	1.89	130
13	6S1P	6S1P-13	12.7	10.68	1.89	1.89	130
14	6S1P	6S1P-14	12.7	10.68	1.89	1.89	130
15	6S1S	6S1S-15	12.7	13.35	1.89	1.89	130
16	6S1S	6S1S-16	12.7	13.35	1.89	1.89	130
17	6S1S	6S1S-17	12.7	13.35	1.89	1.89	130
18	6S1S	6S1S-18	12.7	13.35	1.89	1.89	130
19	3S1P	3S1P-19	12.7	10.35	1.89	1.89	133
20	3S1P	3S1P-20	12.7	10.35	1.89	1.89	133

このブロック高さの情報をもとに高さ制約を満たしながらピースを引き上げていくことになる。ここでは建屋 K1 を例にして、高さ制約遵守のための処理を説明する。K1 では切り出し場側から順に K1-1、K1-2 と定盤に番号を振り、K1-4 が建屋出口側となっている。ここでピース 1～5 を引き上げ終え、次にピース 6 を引き上げようとしている状況を想定する。この状況を Fig.4-16 に示す。

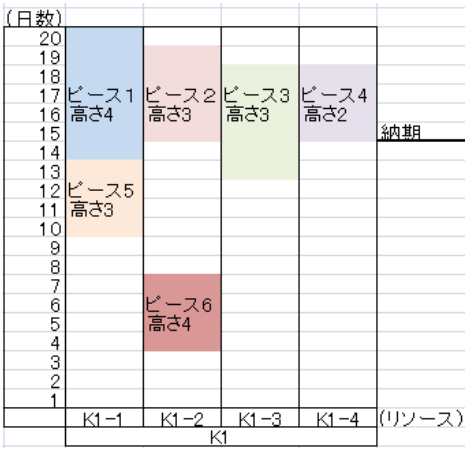

Fig.4-16 ピース 6 を引き上げる直前の Heap 図

ピースを引き上げるためには、まず引き上げる目標となる下側境界線を得る必要がある。高さ制約を考慮しない場合、すでに引き上げたピースの下側境界線とピースの納期を比較して、下にあるほうを選ぶ。この場合 Fig.4-17 に示すような下側境界線になりこれを目標に K1-2 にあるピース 6 を引き上げることになる。

しかし高さ制約を考慮した場合、ピース 6 の作業が完了し出棟する際に、高さ 3 メートルのピース 3 の上を通過することになる。ピース 6 の高さは 4 メートルであるため、合計が 7 メートルとなり建屋の高さ制限に達してしまう。そこで目標となる下側境界線の決定時に、納期だけでなく周囲の高さを考慮して Fig.4-18 のような下側境界線をえることで、高さ制約を満足することができる。

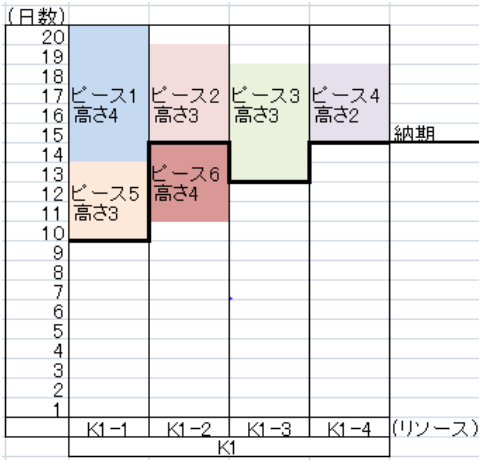

Fig.4-17 ピース 6 を高さ制約を考慮せず引き上げた Heap 図

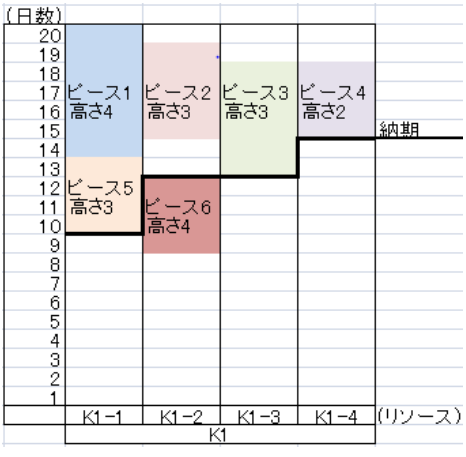

Fig.4-18 ピース 6 を高さ制約を考慮し引き上げた Heap 図

具体的に、Heap 法プログラム上での上記の処理の実装方法について述べる。ピースの引き上げ処理を始める前に、Fig.4-19 に示すような高さ行列と、Fig.4-20 に示すような高さフラグ行列を生成する。それぞれ横軸に対象となるリソースの区画、縦軸に日数

をとる。高さ行列には、Heap 図でピースが存在する場合に、リソースと日数を対応させた位置にそのピースの高さを入力する。高さフラグ行列はすべてを 0 で初期化したのち、対応する位置のうち作業開始日に 1、完了日(出棟日)に 2 を入力する。Fig.4-19 と Fig.4-20 は Fig.4-17 のピースの状況を反映しており、ピース 6 を文字色が赤で示している。

(日数)					
20	4	0	0	0	
19	4	3	0	0	
18	4	3	3	2	
17	4	3	3	2	
16	4	3	3	2	
15	4	3	3	2	
14	4	4	3	0	
13	3	4	3	0	
12	3	4	0	0	
11	3	4	0	0	
10	3	0	0	0	
9	0	0	0	0	
8	0	0	0	0	
7	0	0	0	0	
6	0	0	0	0	
5	0	0	0	0	
4	0	0	0	0	
3	0	0	0	0	
2	0	0	0	0	
1	0	0	0	0	
	K1-1	K1-2	K1-3	K1-4	(リソース)
	K1				

Fig.4-19 Fig.4-17 の状況を反映した高さ行列

(日数)					
20	2	0	0	0	
19	0	2	0	0	
18	0	0	2	2	
17	0	0	0	0	
16	0	0	0	0	
15	0	1	0	1	
14	1	2	0	0	
13	2	0	1	0	
12	0	0	0	0	
11	0	1	0	0	
10	1	0	0	0	
9	0	0	0	0	
8	0	0	0	0	
7	0	0	0	0	
6	0	0	0	0	
5	0	0	0	0	
4	0	0	0	0	
3	0	0	0	0	
2	0	0	0	0	
1	0	0	0	0	
	K1-1	K1-2	K1-3	K1-4	(リソース)
	K1				

Fig.4-20 Fig.4-17 の状況を反映した高さフラグ行列

高さ制約を遵守するために確認が必要なのは、「自ブロックから見て入口側の定盤にあるブロックが出棟日に通過することができるか」と「自ブロックが出棟するときに、出口側にあるブロックの上を通過できるか」の 2 点である。

前者の条件を確認するには次の手順を踏む。まず高さフラグ行列のなかで、自ピースが使用する期間すべてのフラグに対して、自ピースから入口側にある全てのリソースについて、出棟日を表すフラグ 2 の有無を調べる。もしフラグ 2 が存在する場合、高さ行列からそのフラグと同じ位置にある高さを取得し、自ピースの高さと足し合わせ高さ制約と比較する。高さ制約に抵触する場合は、該当のリソースの下側境界線を 1 日下げる。以上の動作を制約に抵触しなくなるまで繰り返す。

後者の条件を確認するには次の手順を踏む。まず高さ行列のうち自ピースの出棟日の行を取り出す。自ピースが存在するリソースより出口側にあるピースの高さと、自ピースの高さを各々足し合わせる。その合計が高さ制約の値以上になるリソースが一つでもあれば、該当リソースの下側境界線を 1 日下げる。以上の動作を制約に抵触しなくなるまで繰り返す。

以上の両条件を確認し、両方の条件が満たされれば、そのときの下側境界線を目標日として自ピースを引き上げる。

このような高さ制約に関する処理をあらたに加えて、Heap 法によるプル型スケジューリングプログラムを当該造船所の実データに適用した。

4.3 Heap 法の適用

当該造船所のブロック組立計画に対して、Heap 法によるプル型スケジューリングを適用した結果を述べる。まず現行のブロック組立計画データにしたがって、現行計画を Heap 図で表したものを改めて、Fig.4-21 に示す。またガントチャート表示したものを Fig.4-22 に示す。各色の意味する作業は Table 4-3 と等しい。前章の場合と同様に、現行計画のアイドル期間を算出する。当該造船所では造船所全体をモデル化したため、あるピースで表された作業工程iの完了日y_i、次工程の作業開始日をx_iとしたとき、アイドル期間の定義を

$$F_{idling\ time} = \sum_{i=1}^{N} (x_i - y_i) \tag{4.2}$$

とする。

このとき、現行計画のアイドル期間を算出すると、1280 日となった。外注ブロックの完成予定日が遅延した場合や、現場で直前に定盤を決定するといった計画作業の遅れ、部材作成の前倒し等の理由で、アイドル期間が増加している。

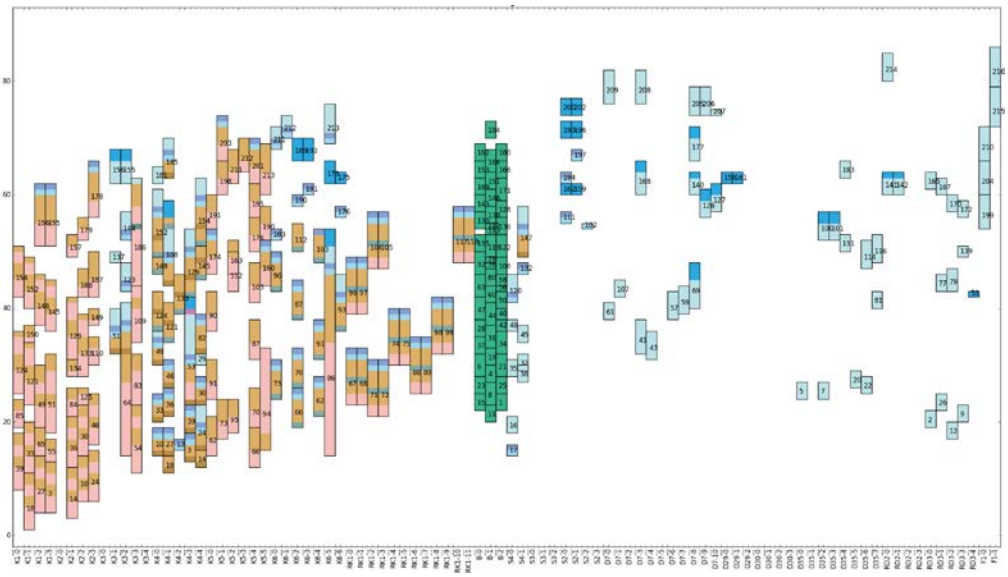

Fig.4-21 現行計画の Heap モデル図

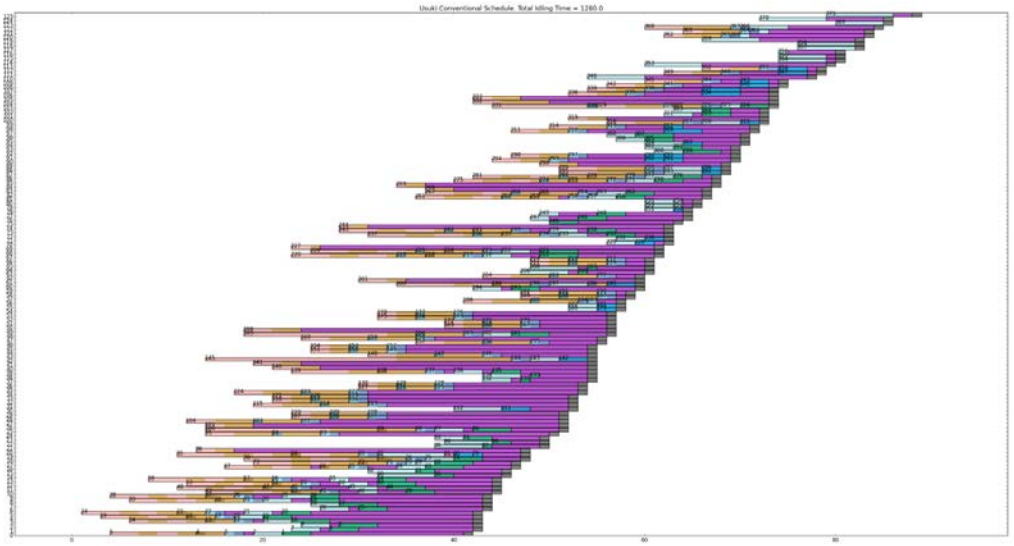

Fig.4-22 現行計画のガントチャート

この現行計画に対して、ブロック製作順序をそのままピースを引き上げる順番、使用する定盤を同じとし、また一時的に高さ制約を解除した状態でプル型スケジューリングを適用した結果のうち Heap 図を Fig.4-23 に、ガントチャートを Fig.4-24 に示す。アイドル期間は 633 日に削減し、リードタイムが 2 日短縮された。Fig.4-21 と Fig.4-23 を比較すると、K1 から K6 までの組立工程において各ブロックの作業の間にあった隙間時間が詰まりアイドル期間を削減している。しかし、現行計画の時点で塗装工場が全て常時稼働するほどの物量が流れているために、ピースを引き上げようにも隙間時間がなく、リードタイムの短縮のボトルネックとなっていることがわかる。

Fig.4-22 と Fig.4-24 に示すガントチャートを比べると、アイドル期間を表す紫のバーが大きく減っていることが見て取れる。現行計画では、おおよそ納期の順にブロックの制作が始まっている。しかしプル型スケジューリングの適用後は、現行計画の開始順に合わせた順序でピースを引き上げたにも関わらず、納期の遅いブロックを早い日付で製作開始する計画となっていた。これは、制作に長期間かかるブロックはこれ以上ピースを引き上げる隙間がないため開始日が他に比べて早くなっている点や、部材の入荷日を早めに設定しているためにアイドル期間が増えていたものを、ジャストインに入荷するようスケジューリングしたためと考えられる。

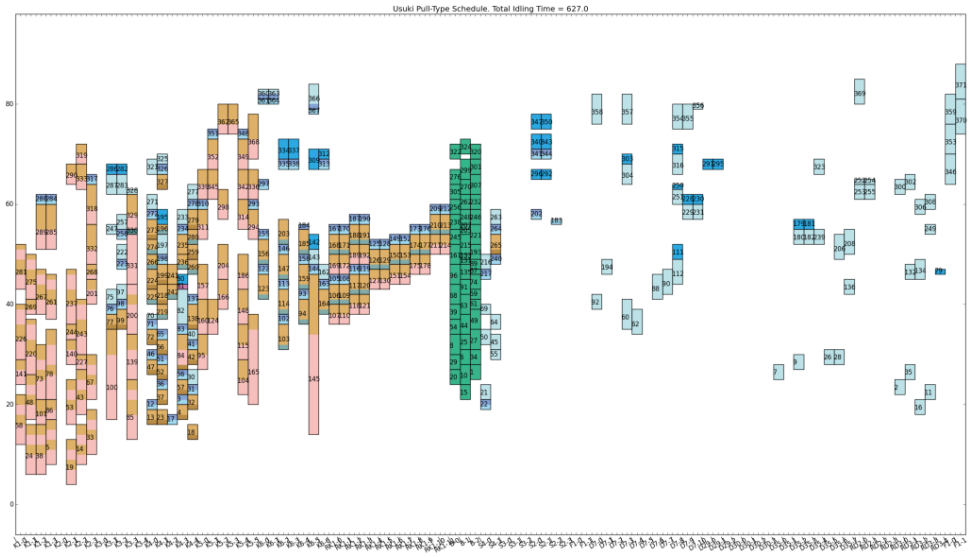

Fig.4-23 プル型スケジューリング後 Heap 図(高さ制約未考慮)

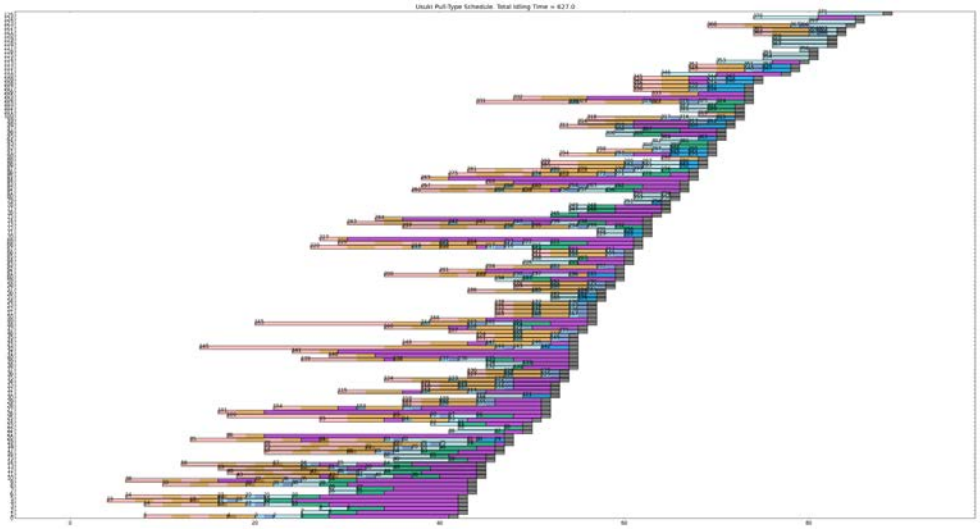

Fig.4-24 プル型スケジューリング後ガントチャート(高さ制約未考慮)

4.4 塗装工場の改善検討

Heap 法の適用によって、塗装工場の作業量の多さが当該造船所のボトルネックとなっており、工程計画においてリードタイム短縮のためには、ここになんらかの改善が必要なことが明らかとなった。ここで造船所のスタッフと協議のもと、現在通常勤務時間のみ稼働している塗装工場が、夜間作業も可能となった場合に工程計画にどのような影響が出るか検討を行った。

塗装工場では 2 日から 4 日の間、ブラストおよび塗装作業が行われる。ここで夜間作業が可能となったことで

Case 1: 塗装作業期間が 4 日以上のものを 1 日短縮する

Case 2: 塗装作業期間が 3 日以上のものを 1 日短縮する

の 2 つの場合を仮定する。

Case 1 では、塗装作業期間が 4 日のものが夜間作業によって 3 日で完成するようになった場合を想定している。Case 2 は作業期間 4 日のものが 3 日で完成するようになった場合に加え、作業期間 3 日のものが 2 日で完成するようになった場合を想定している。両ケースに備え、ブロックデータの該当部分を修正したものを作成し、そこからピースを生成した。

ここでの改善検討では、投入順序および作業区画は現行計画と同じものになるように設定している。また屋内建屋 K1 および K2 の高さ制約は本検討では未考慮としている。

Case 1 の場合について、Fig.4-25 にプル型スケジューリング前の Heap 図、Fig.4-26 にプル型スケジューリング前のガントチャート、Fig.4-27 にプル型スケジューリング後の Heap 図、Fig.4-28 にプル型スケジューリング後のガントチャートを示す。Case 1 ではアイドル期間が 585 日に削減され、リードタイムが 6 日間短縮された。改善前の現行計画 Fig.4-21 と Case 1 の現行計画 Fig.4-25 を比較すると、塗装工場で作業を行うブロック間において、作業期間短縮のために隙間時間が発生していることがわかる。Fig.4-26 を見ると、プル型スケジューリングの適用によって、作業期間短縮によって生まれた隙間を詰めることができ、それにより K1 や K4 での作業にあった隙間を詰めることができている。その結果、リードタイムを 6 日間短縮することに成功している。

Case 2 の場合について、Fig.4-29 にプル型スケジューリング前の Heap 図、Fig.4-30 にプル型スケジューリング前のガントチャート、Fig.4-31 にプル型スケジューリング後の Heap 図、Fig.4-32 にプル型スケジューリング後のガントチャートを示す。

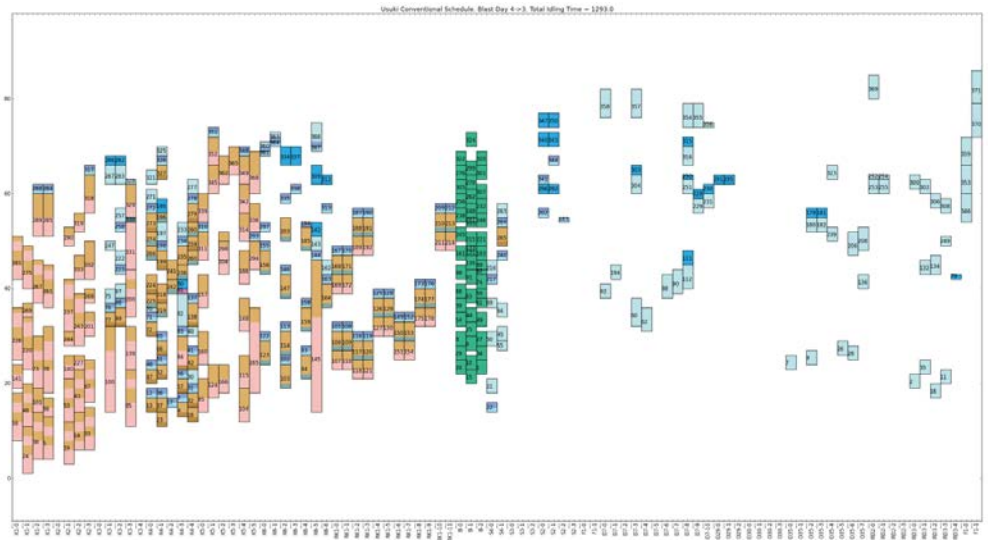

Fig.4-25 Case 1 を反映したプル型スケジューリング前の Heap 図

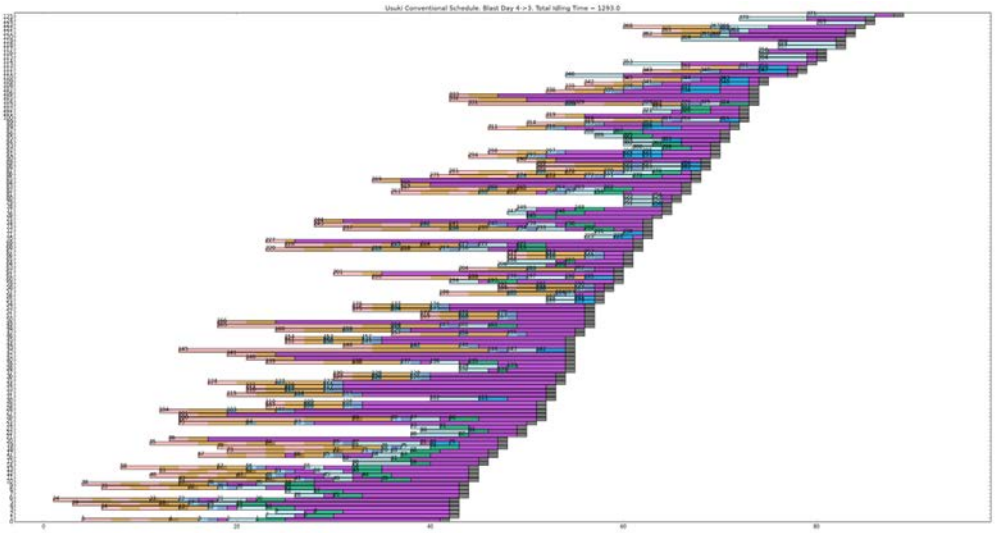

Fig.4-26 Case 1 を反映したプル型スケジューリング前の Heap 図

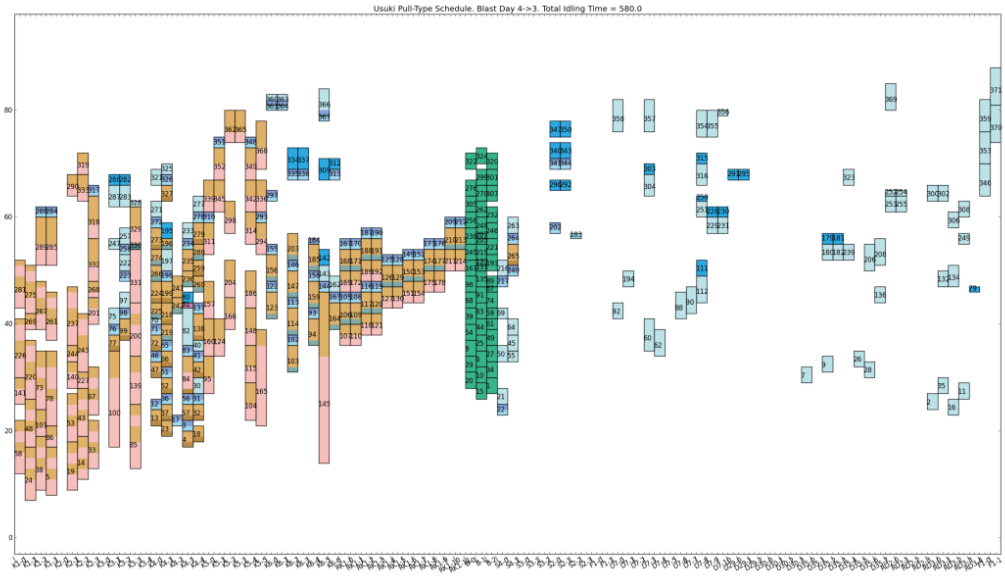

Fig.4-27 Case 1 を反映したプル型スケジューリング後の Heap 図

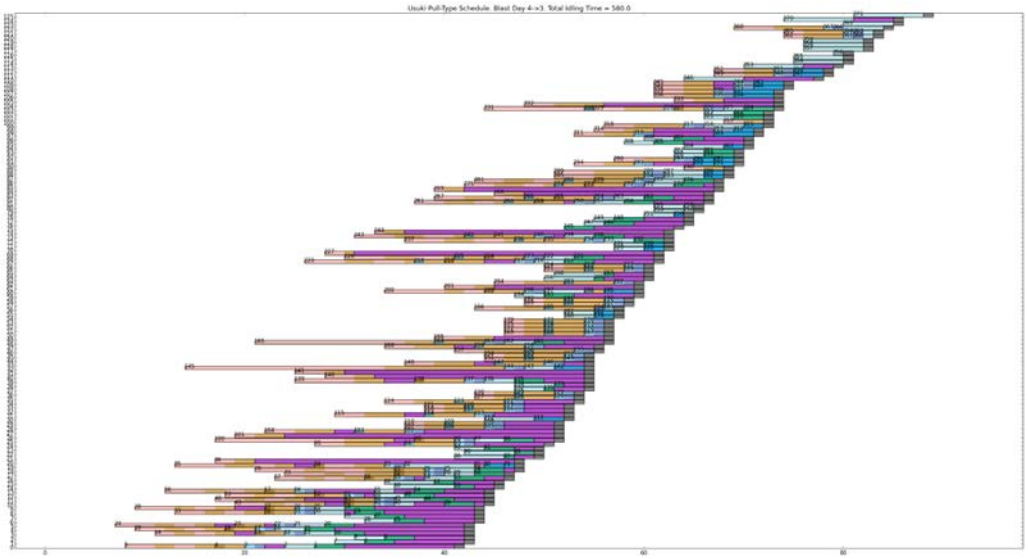

Fig.4-28 Case 1 を反映したプル型スケジューリング後のガントチャート

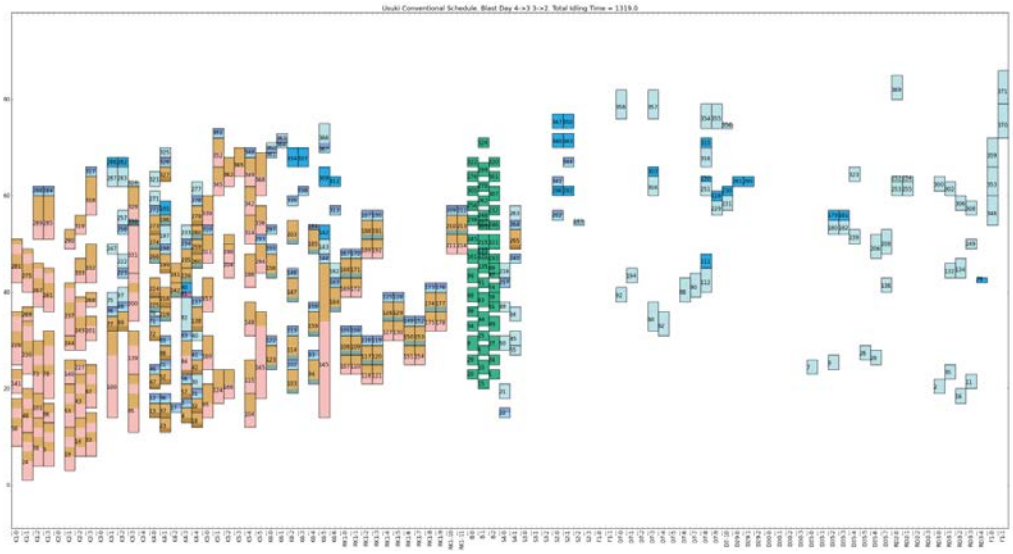

Fig.4-29 Case 2 を反映したプル型スケジューリング前の Heap 図

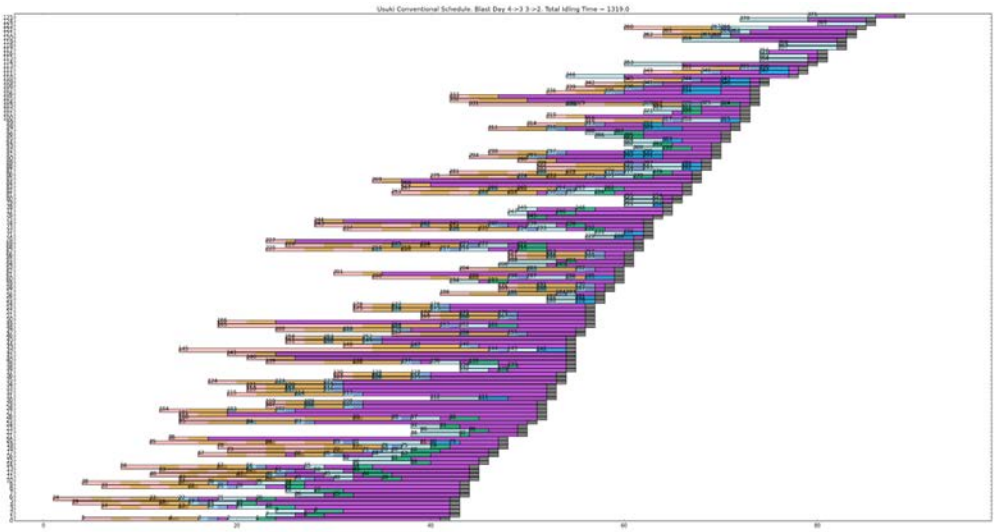

Fig.4-30 Case 2 を反映したプル型スケジューリング後の Heap 図

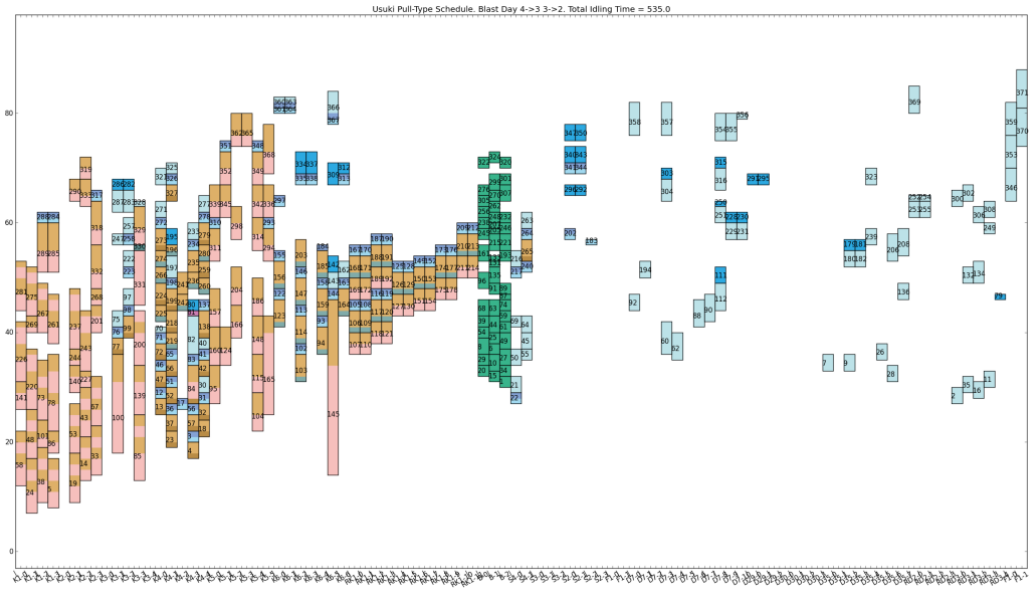

Fig.4-31 Case 2 を反映したプル型スケジューリング後の Heap 図

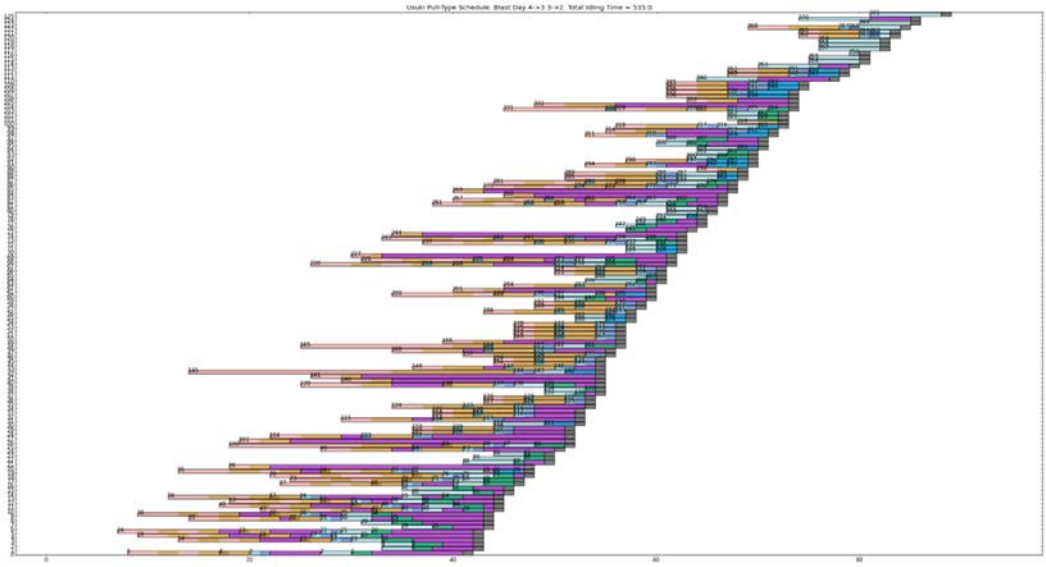

Fig.4-32 Case 2 を反映したプル型スケジューリング後のガントチャート

Case 2 ではアイドル期間が 538 日に削減され、リードタイムが 6 日間短縮された。Case 1 の現行計画 Fig.4-25 と Case 2 の現行計画 Fig.4-29 を比較すると、塗装作業が更に短縮された結果、工場の隙間時間が増加している。Fig.4-31 に示すプル型スケジューリング適用後の Heap 図をみると、隙間時間を詰めてアイドル期間を削減したものの、作業時間の長いブロックが連続して塗装工場の同じ建屋に入ったことや、次工程までの待機時間がなかったために引き上げられなかったピースがあることが影響し、塗装工場にところどころ隙間時間が発生している。また、K4 での作業が詰まり始めた結果、K1 か

らK3の組立工程において、隙間時間を詰めきれていないことがわかる。

以上のことから、現状の投入順序および定盤計画の上では、塗装工場の夜間作業が可能になったとした場合、リードタイムの短縮を考えるとCase 1の場合が最適となり、アイドル期間の削減を考えるとCase 2が最適となる。Case 2に関しては定盤計画を変更し、使用するリソースを適宜調整することでさらなるリードタイムの短縮が考えられる。

4.5 アイドル期間を最小化する定盤選択の検討

これまでの検討では使用する定盤は現行計画と同じ定盤を選択していた。ここでは定盤計画の自動化のために、最適化手法である焼きなまし法によって、各ピースが使用するリソースを変更しつつアイドル期間を調査し、アイドル期間を最小化する定盤計画を検討する。

この検討では、塗装作業の改善などを行っていない初期のブロック組立データを使用する。ピースを引き上げる順序はこれまでと同様に現行計画のブロック作製順序を踏襲している。またK1およびK2の高さ制約を遵守する。焼きなまし法による定盤計画最適化を行うリソースはK1からK6の範囲とする。この理由として、塗装工程のリソース使用状況をみると、隙間なく詰まっておりリソース変更の効果が薄いと考えられること、また海上のバージのうち事前に使用することがわかっているブロックが少なく、大多数は計画後の蔵置で使われるため、計画中に検討することができないことの2点が挙げられる。

焼きなまし法における近傍解の生成方法は、ランダムに選択したブロックの該当する工程について、使用する定盤を左右に一つ、ずらす処理をすることで近傍の解を得る方法を用いる。

以上の条件のもと、焼きなまし法をもちいた定盤計画最適化を行った。目的関数であるアイドル期間の計算回数による変化をFig.4-33に示す。縦軸にアイドル期間をとり、横軸に温度を冷却した回数を示している。計算によるアイドル期間の変化を青線で示し、初期解のアイドル期間の値である657日を赤線で示している。

初期解は現行計画の定盤計画を用いており、焼きなまし法の特徴通り計算の初期では目的関数の値が悪化する解も受け入れるために、アイドル期間が大幅に増加している。計算が進むに従って、比較して良くなった解を受け入れるようになるため、アイドル期間が減少していく。冷却回数が100回前後でアイドル期間の初期値を下回り、計算が進むにつれてゆっくりとアイドル期間が減少している。最終的に得られたアイドル期間の最小値は344日となり、初期値からおよそ半減する結果を得られた。

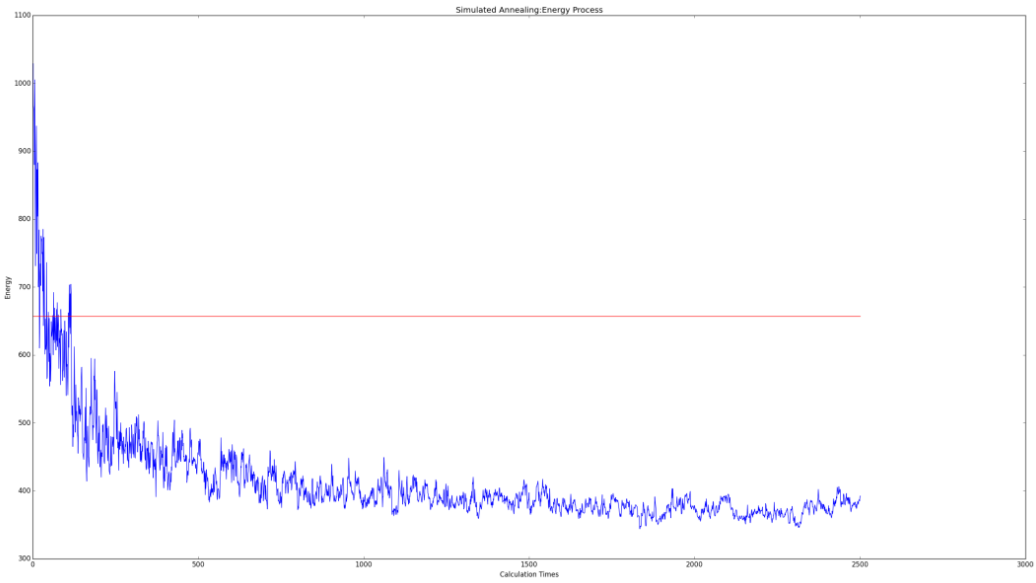

Fig.4-33 計算経過によるアイドル期間の遷移グラフ

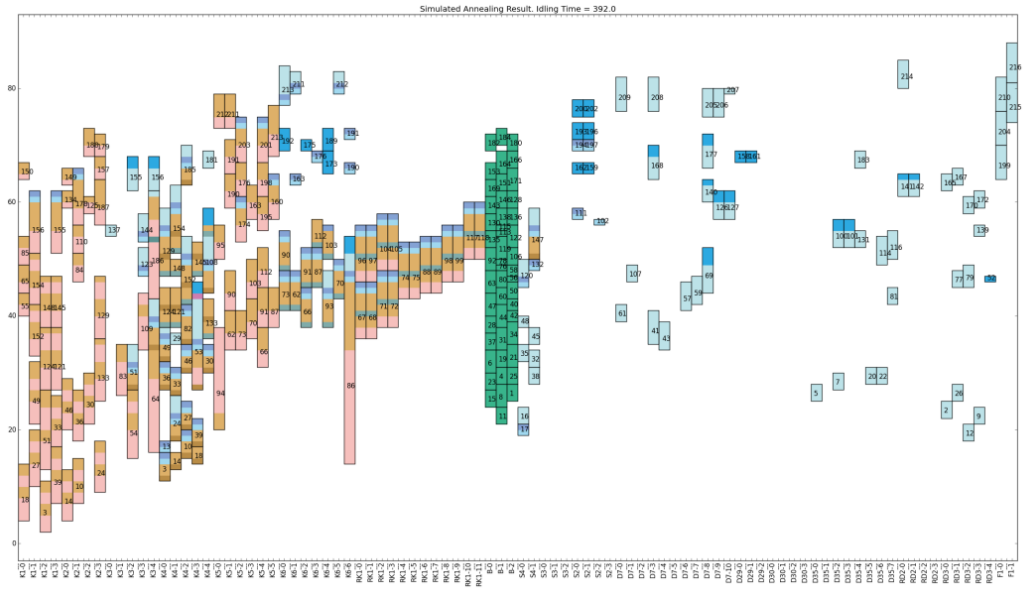

Fig.4-34 焼きなまし法により得た計画の Heap 図

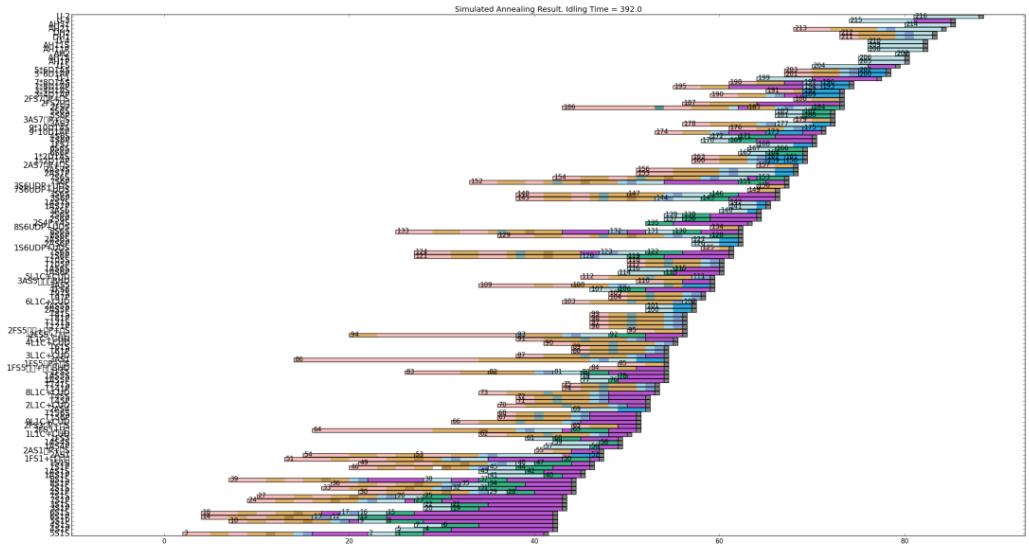

Fig.4-35 焼きなまし法により得た計画のガントチャート

焼きなまし法によって得られたアイドル期間最小の結果のうち、Heap 図を Fig.4-34 に、ガントチャートを Fig.4-35 に示す。計算時間は約 20 時間となった。前述のとおり、アイドル期間は 344 日となり、リードタイムは 2 日短縮された。アイドル期間は大きく削減されたが、リードタイムの短縮日数は、単純にピースを引き上げた場合の結果である Fig.4-23 と同じ値となった。これは塗装工場がすでに連続稼働しなければならないほどの仕事量となっているため、ここが支持棒のように引き上げる障害となっているためである。また K1 と K2 の高さ制限のために各ブロックの作業のあいまに、隙間時間が発生してもピースを引き上げることができないことが考えられる。

K1 から K6 について、定盤計画の変更の結果、現行計画の定盤計画では早い日数にあったピースが、納期側へ引き上げられている。これは現行計画の段階では同じ定盤で組み立てる予定だったピースが別の定盤で作られるようになったことで、引き上げる障害がなくなり、上に移動しやすくなったためだと考えられる。

4.6 考察

造船所 B に Heap 法を適用し、定盤計画の自動化による工程計画の最適化を行った。現行計画に対して Heap 法によるプル型スケジューリングを適用するとアイドル期間 1280 日から 633 日まで削減し、リードタイムを 2 日短縮することができることを確認した。この結果、塗装工程が常に連続稼働しているためにリードタイム短縮の障害となっていることが明らかになった。そのため塗装工程に対する改善とその効果を検証した。塗装工場が夜間作業を行うことができる場合を想定し、2 つのケースを仮定した。

Case 1 は塗装作業が 4 日かかるものを 3 日でこなせるようになった場合である。こ

の場合、塗装工程に隙間時間が生まれた結果、プル型スケジューリングによりアイドル期間が 585 日まで削減されリードタイムが 6 日短縮した。Case 2 は Case 1 に加えて塗装作業が 3 日かかっていたものが 2 日でこなせるようになった場合である。この場合、アイドル期間は 538 日と更に削減されたが、リードタイムは 6 日短縮と変化がなかった。これは塗装工程に隙間時間が生まれたものの、定盤計画の関係からその隙間を完全に埋めることができなかったことや、作業区画 K4 が混雑しはじめ、ピースを引き上げることができないものが現れはじめたためである。

以上の結果から、塗装作業の改善には大きな効果が現れるものの、塗装作業の改善だけでなく、その前後の工程の改善を行う必要があることが明らかになった。本検討では定盤計画を考慮していないため、各工程で作業が行われる区画を変更することでさらなる効果を得られると考えられる。

次に焼きなまし法を利用して、定盤計画の自動化を行った。このとき一部建屋の高さ制約を考慮してブロックを組み立てるように注意し最適化計算を行った。この結果、アイドル期間を 344 日まで削減し、リードタイムを 2 日短縮した。これにより定盤計画を考慮することがアイドル期間を大きく削減することに多大な影響を与えることがわかった。反面リードタイムの短縮については塗装工場の作業量の問題や、高さ制約の問題が絡み、定盤計画だけでは解決に至らないことが明確となった。

この焼きなまし法による結果を整理し、造船所の組立計画立案を行う部署にて検討していただいたところ、ブロック組立作業の順番のミスやいくつか見つかった。また本来あるブロックの組立に利用するために別の定盤で作製され運ばれる部材の完成のタイミングがずれているために、実行不可になっている計画があることがわかった。これらについては、ブロック組立順序のさらなる調査と、ピース定義のためのデータファイルの生成をより簡易的にするなど、計画手法適用の改善を図る必要性がある。

今後の課題としては、定盤計画だけでなくブロックの制作順序の最適化を行うことや、塗装作業が改善された場合に更に定盤計画の最適化を行うこと、塗装工場が増設された場合の効果検討、得られたブロック組立工程の実用化に向けた修正などが考えられる。

第 5 章 工程計画 Web アプリケーションの提案

5.1 Web アプリケーションの概要

これまで、造船所 A および造船所 B に対して Heap 法によるプル型スケジューリングを適用し、アイドル期間を削減したブロック組立工程計画の検討を行ってきた。これらの計画を作成するためには、数値計算言語である MATLAB やプログラミング言語 Python を用いて計算を行っていた。しかし、この計画手法を実際に造船所で作業者が使用することを考えると、計画手法に関する知識だけでなくプログラム言語に関する専門知識を必要とし、利便性に欠ける。反面、造船所では組立工程計画や定盤計画の作成に Microsoft の表計算ソフトである Excel を使用して手作業によって計画作成を行っている。Excel での計画作成は、他の作業者と作成環境を合わせやすく、Excel シート上のガントチャートを操作することで調整することができる利点があるが、設備制約や同期制約を満足させるような組立工程計画を作成することが難しい。

本研究では、Heap 法を造船所で実際に使える手法にするために、Heap 法に基づく WEB アプリケーションツールの開発[14]を行った。WEB アプリケーションとは、WEB ブラウザ上で動くアプリケーションの総称である。WEB ブラウザに代表されるクライアントからの入力をリクエストとして WEB サーバへ送信し、計算などの処理をサーバ上で行ったのち、結果をレスポンスとして WEB ブラウザへ送信する。この概要図を Fig.5-1 に示す。

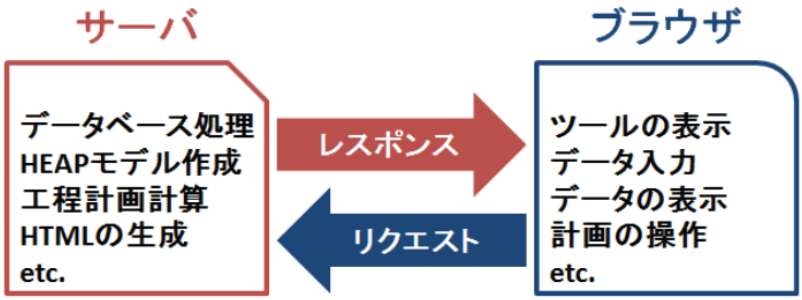

Fig.5-1 WEB アプリケーションの概念図

本研究で開発する WEB アプリケーションでは、WEB ブラウザ上に表示した計画ビューワで組立工程に関する情報の登録や、ブロックデータの登録、計算結果から得られた組立計画の編集を行う。サーバ上では、ブラウザから送られてきた情報をデータベースへ登録したり、そのデータをもとに Heap 法によるスケジューリングを行う、また得られた結果をブラウザで表示するためにデータを整形する、といった処理を行う。

5.2 開発環境

WEB アプリケーションを開発する環境について説明する。サーバ上での処理を行うためのプログラムについてはプログラム言語 Python をもちいて構築した。Python には WEB アプリケーション開発を支援するフレームワークとして Django がある。これはサーバとブラウザ間の通信を行うためのプロトコルや、データベースの操作に必要な SQL などを Python から操作しやすい関数にして提供している。

ブラウザに表示するスケジューリングツールについては、HTML5、CSS3、JavaScript を用いて構築する。HTML5 は、マークアップ言語 HyperText Markup Language のことで 5 回目の改訂版のことである。HTML はウェブ上に文書を記述するための言語であり、「要素」と呼ばれる機能で単語や段落を囲むことで、文書の装飾や表示を行うことができる。CSS とは Cascading Style Sheets のことで、HTML の要素の装飾を指示するための仕様のことである。これにより HTML 単体以上の表現を実装することができる。JavaScript とは、プログラミング言語のひとつで、ウェブブラウザに実行環境が実装されており、ウェブサイトに動的な処理を追加することができる。これによりユーザーインターフェースの開発をすることができる。　本研究では CSS に Twitter 社から提供されている Bootstrap を使用した。これは CSS を用いた複雑な表現を、簡単な表記で実装することができるものである。また JavaScript についてはフレームワークである jQuery および jQueryUI を用いて実装した。このフレームワークはインターフェースによく用いられる処理を簡易な表記で実装することができるものである。

以上の環境のもと WEB アプリケーションの開発を行った。

5.3 サーバ側システムの構築

サーバ側のシステム要件について述べる。サーバ側で最低限実装する機能は

1. CSV ファイルの展開とデータベース登録
2. Heap 法による工程計画計算
3. 工程計画の編集結果の保存

である。

この WEB アプリケーションツールでは、工程情報やリソース情報、ブロック組立データを CSV ファイル形式を用いる。CSV とは Comma-Separated Values の略で、項目をカンマで区切ったテキストデータおよびテキストファイルのことである。これは表計算ソフトやデータベースなどでデータを交換する際の標準的なファイル形式で Excel から開くこともできる。造船所で主に使われるソフトが Excel であるためにこの

形式を採用した。リソースを定義するための CSV ファイルの例を Table 5-1 に、ピースデータを定義するための CSV ファイルの例を Table 5-2 に示す。

Table 5-1 リソース定義 CSV ファイル

Table 1 Example of CSV file for resource parameter.

No.	Name	Layer	Resource ID	Capacity	Resource Name	Selection Rule
1	Shipyard 1	1	1	4	Assembly Line 1	0
1	Shipyard 1	1	2	4	Assembly Line 2	0
2	Shipyard 2	1	1	5	Assembly Line 1	0

Table 5-1 に示すリソース定義ではリソースデータ番号(No.)、リソースデータ名(Name)、計算順序を指示するレイヤー(Layer)、リソース ID(Resource ID)、リソースの許容量(Capacity)、リソース名(Resource Name)、リソース選択ルール(Selection Rule)を定義する必要がある。この例では、Shipyard 1 と Shipyard 2 の 2 つの造船所のリソースを定義している。Shipyard 1 は 2 つの組立工程を持っている。この組立工程 Assembly Line 1 と Assembly Line 2 をリソースとして考える。各組立工程は作業区画を 4 区画持っている。これを許容量として定義する。Shipyard 2 は 1 つの組立工程を持っている。こちらも同様に考える。両造船所は独立しているためどちらから計画を立てても問題ない。そのためレイヤーを 1 としている。計劃順序がある場合は、先に計算したいものに若い番号を振ることで対応する。リソース選択規則は各作業区画のどれを選択するのかについて指示を与えるもので、事前の指示どおり選択する場合は 0、輪番方式に選択する場合は 1、本研究で提案したリソース選択規則 Rule 3 を利用する場合は 2 を入力する。

Table 5-2 ピース定義 CSV ファイル

Table 2 Example of CSV file for pieces data.

No.	Name	Piece ID	Link	Due Date	Job ID	Job Name	Workload	Resource ID	Resource	Sub Resource	Lower Border	Upper Border
1	Block 1	s1-b1	0	15	0	艤装	0	1	1	1	10	13
2	Block 2	s1-b2	1	13	1	溶接	0	1	2	2	8	10
3	Block 3	s2-b3	0	10	1	取付	0	2	1	1	4	7
3	Block 3	s2-b3	0	10	2	溶接	0	2	1	1	7	9

Table 5-2 に示すピース定義は、ピース番号(No.)、ピース名(Name)、ピース ID(Piece ID)、接続(Link)、納期(Due Date)、仕事 ID(Job ID)、仕事名(Job Name)、作業人員数(Workload)、リソース ID(Resource ID)、使用リソース(Resource)、選択リソース(Sub Resource)、下側境界線の値(Lower Border)、上側境界線の値(Upper Border)を定義する必要がある。例としてピース番号 3 番に注目する。ピース 3 番は Block 3 の組立工程を表すピースである。ピースの ID は s2-b3 であり、この ID が同じであれば同じピースであると処理する。ピースの接続先は 0 で存在しないため、このピースより先に引き

上げる必要のあるピースは存在しない。このピースの納期は第 10 日目である。このピースは取付と溶接の 2 つの仕事で構成されている。作業人員数は今回考慮していない。ピース ID が 2 であるため Table 5-1 の No.2 のリソースで作られることを意味する。使用リソースは 1 のため、Table 5-1 のリソース ID が 1 を使用し、選択リソースは 1 であるため作業区画 1 番を使用する。取り付け作業を 3 日間おこなったあと、溶接作業を 2 日間行う。
以上の CSV ファイルを展開し、計算処理に必要な形式に編集しデータベースへ保存する処理を実装した。

次に Heap 法による工程計画計算の処理を実装した。これに関しては、造船所 B へ Heap 法を適用した際のプログラムが Python で記述されていたため、サーバ内での処理に適した形に修正して実装した。この工程計画計算処理にて得られた計画データはデータベースへ保存するようになっている。

最後に、ブラウザで編集された計画データが送られてきた際に、それらをデータベースへ保存する処理を実装した。

サーバ側でのシステム構築は主に以上の 3 つを実装した。

5.4 ブラウザ側システムの構築

ブラウザ側のシステム要件について述べる。ブラウザ側で必要となる処理は、

1. 新規計画の作成および CSV 送信フォーム
2. 工程計画の編集機能および保存機能
3. ピースの詳細情報閲覧ページ

が考えられる。

新規計画の作成と CSV ファイルの送信フォームに関する処理は HTML の基本的な構文で実装した。

工程計画の編集機能および保存機能については、JavaScript を用いて実装した。JavaScript を利用することで HTML により記述されたパーツを動かしたり、HTML 構文に設定する値を操作したりすることができる。この動作を利用し、HTML と CSS で記述したピースを移動してリソースを変更する、使用するリソースが変更されないように固定するといった機能を実装した。また投入順序を手動で並び替える機能も同様に実装した。

Heap 図の表示画面では、ピースによって各作業の作業時間や使用リソースを明示できるが、そのピースの詳細な情報を閲覧することができない。そこで各ピースをクリックすることで、そのピースが持つ情報を表示して、作業内容や納期を確認できるページ

を実装した。

5.5 実工程データへの適用

4 章で扱った造船所 B を例にして、本研究で開発した WEB アプリケーションを説明する。Fig.5-2 に WEB アプリケーションのトップ画面を示す。

Fig.5-2 WEB アプリケーショントップ画面

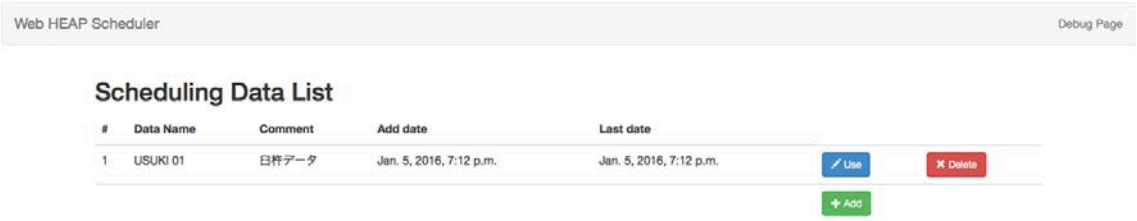

Fig.5-3 計画データリストの表示画面

トップ画面のスタートボタンをクリックすると、Fig.5-3 に示すような計画データのリスト画面に遷移する。このリスト画面には WEB アプリケーションに登録したスケジューリングデータが一覧として表示される。例えば 4 章で扱った造船所 A の場合、このデータ名に「造船所 A」として登録し、このスケジューリングデータの中に、全 13 工程の各計画結果が紐付けられることになる。Fig.5-3 には造船所 B の名前とそのデータに関するコメント、登録日、最終編集日が表示されている。このスケジューリングデータを閲覧・編集する場合は「Use」ボタンをクリックしてスケジューリング画面へ遷移する。またデータを削除したい場合は「Delete」ボタンをクリックすることで削除できる。また新規のスケジューリングデータを入力する場合は、「Add」ボタンをクリックすることで Fig.5-4 に示すような新規登録画面に遷移することができる。

Fig.5-4 新規計画データ登録画面

新規登録画面では、スケジューリングデータの名前、このデータに関するメモ、リソース定義 CSV ファイルを入力する必要がある。ここのリソース定義 CSV ファイルではこのスケジューリングで扱うすべてのリソースを定義しておく必要がある。4 章で扱った造船所 A であれば大組・中組ブロック組立工程の計 13 工程の各リソースを登録し、5 章で扱った造船所 B であれば造船所を一つの工程とみなしてリソース定義したものを登録する。

新規登録画面で必要な入力を行いサーバへ送信すると、リソースの項目以外情報のないスケジューラー画面が表示される。その画面の指示にしたがってピースデータを入力すると、そのピースデータをもとにプル型スケジューリングを行う。その後 Fig.5-5 に示すように、計画結果の Heap モデル図が表示されたスケジューラー画面が表示される。

Fig.5-5 スケジューラー画面(製作順序リスト表示)

スケジューラー画面では、左側にこのスケジューリングデータのリソースと、現在選択しているリソースのもとでスケジューリングを行うピースの一覧が表示される。「Order」のタブではピースを引き上げる順序を表示しており、上から順にピースを引き上げる。もしも引き上げる順序を変更したいピースがあれば該当のピースの欄をドラッグ操作で上下に移動することで実行できる。「Piece」のタブにはピースの一覧が表示されており、Fig.5-6 の左側のように各ピースに割り振った番号順に並んでいる。

右側には Heap モデル図の表示欄があり、縦軸に日数、横軸にリソースを取り、ピースが表示される。各ピースはマウスをもちいたドラッグ操作で動かすことができ、例えば K4 の 2 番目の定盤で作る予定のブロックを K4 の 1 番目の定盤で作るように変更したい場合は、K4-2 にある該当のピースを K4-1 へドラッグ操作で動かせば良い。また再計算をする場合に必ずその定盤を使うようにしたい場合は、そのピースをダブルクリックするとピースの色が青から赤に変更され、赤の状態であればそのリソースに固定する。

右側の表示画面のうち「Gantt Chart」のタブをクリックすると Fig.5-7 に示すような、ガントチャート画面に遷移する。この画面では縦軸にブロック名、横軸に日数を取り、各ブロックの製作に関わる作業を連続して表示している。Heap モデル図において、あるピースの開始日を早めた場合、ガントチャートの表示にも反映される。計画の微調整をする場合は Heap モデルでリソースを管理しつつ開始日を変更することで、自動的に設備制約を満たしつつ調整をかけることができる。

さらに「Piece Detail」タブをクリックすることで、Fig.5-8 に示すようなピースの詳

細情報を確認できるページに遷移する。左側のリストに表示されたブロック名やピース名のリンクをクリックすると、該当するピースやブロックの詳細情報が表示される。

Fig.5-6 スケジューラー画面(ピースリスト表示)

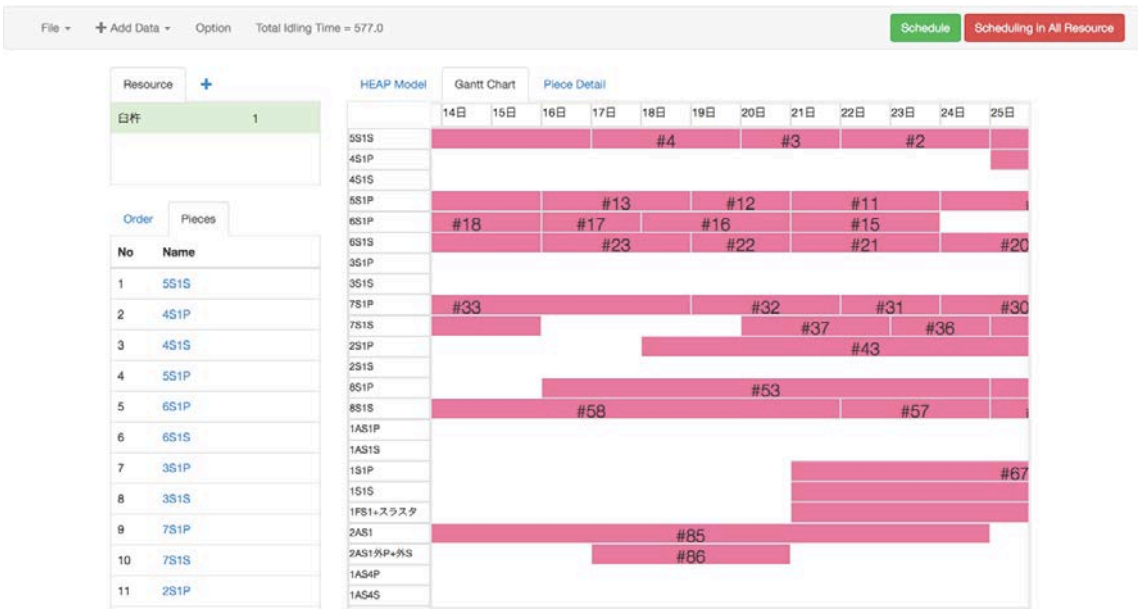

Fig.5-7 ガントチャート画面

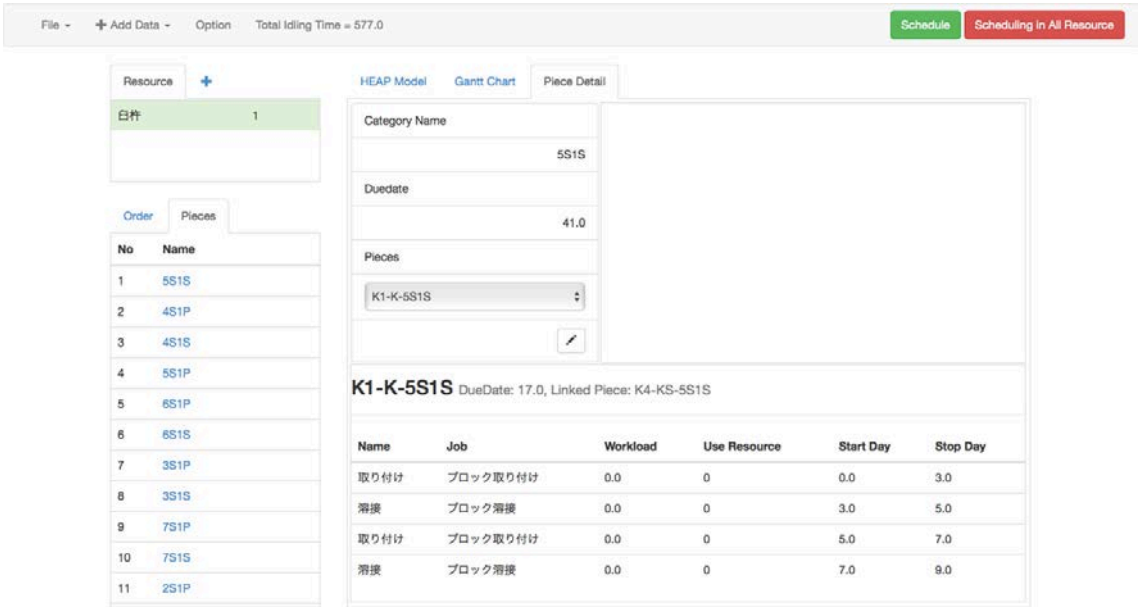

Fig.5-8 ピース詳細情報画面

例えば 5S1S のブロックの組立工程を表すピースのひとつ K1-K-5S1S というピースは取付と溶接を 2 回繰り返す作業であり、開始日を 0 としたときのそれぞれの作業期間や納期などを確認することができる。

以上のように、WEB アプリケーションによるスケジューリングツールとして必要な機能を実装した。

5.6 考察

今回開発した WEB アプリケーションを、造船所作業者の方に紹介し、評価をいただいた。好意的に評価された点を以下に示す。

- WEB アプリケーションの形をとったことで、リソースの定義ができれば対象によらずにスケジューリングが可能である
- 紙面上に出力した Heap モデル図と比較して、各ピースの情報を調べやすい
- ピースを配置するリソースを変更した場合の、工程計画への影響を調べやすい

また、改善点について以下に示す。

- リソースやピースの定義に知識と技術が必要である
- 定盤計画最適化の結果を入力することで、複数の計画の比較ができると良い
- 完成した結果の数値データを出力して、Excel などで読み込めると、作業者へ配布する資料を作りやすい
- 人員や工数の情報がさらに詳しく表示でき、配員作業や平準化ができると良い

今回、Heap 法によるプル型スケジューリングを汎用化し、造船所で活用しやすいツールとなることを目的として、WEB アプリケーション化を行った。これによりユーザーが計画の手法や数値計算のような理論的な部分を意識せずに、直感的な操作で Heap 法を扱うことができる。またデータベースの特徴を活かしたピースの情報の検索と表示機能によって、紙面上や Excel シート上での工程計画に比べて多くの情報を管理しつつ計画を立てることができる。

本 WEB アプリケーションでは、情報の入力形式に CSV 形式を用いた。これは一般的に広く使われる Excel のような表計算ソフトを意識して採用したが、実際に実データからピース定義ファイルを作成してみると入力ミスが多く発生した。表計算ソフトでの入力は扱いやすい反面、人間が関わることで読み間違いや打ち間違いが発生することで、計画作成の障害となることがわかった。また実際の工程計画では作業時数の見積もりと配員を行い、造船所の能力に見合った計画を立てる必要があるが、今回はデータの入力のみ用意し、人員平準化処理を行っていない。

以上より、今後の課題としては、人員平準化を見据えた工程計画処理の追加や、より簡易にリソース定義やピース定義を行えるツールを開発しつつ、WEB アプリケーションの細かい修正を行っていく必要がある。

第 6 章 結論

本研究では、造船所の生産性向上を目的としたブロック組立工程計画の効率化のために、新しい工程計画手法の提案とその実問題適用を行った。そのために研究対象の複数の造船所の工程計画を調査し、ブロック組立工程計画の定式化を行った。それにより明らかとなった制約条件を満たす計画手法として、Heap 法によるジャストインタイム計画手法を提案した。

Heap 法による計画手法は、組立工程を通過するブロックに対して行われる各種作業を、扱いやすい簡単なモデルによってピースとして表現し、そのピースを積み重ねるもしくは引き上げるという単純な処理で工程計画を行う手法である。これによりブロック組立工程計画において考慮する必要のある、設備制約問題と同期制約問題を簡単且つ明確な形で満たしつつスケジューリングを行うことが可能となった。

本研究で提案した Heap 法を実際のブロック組立データに対して適用することで、その実用性を確認した。

造船所 A については、大組ブロック組立工程ラインと中組ブロック組立工程あわせて 13 ラインに対して適用した。その結果大組ブロック組立工程計画の各ラインについて、設備制約を満たしつつ、アイドル期間を大幅に削減可能であることを確認した。各サイクルにおける大組工程全ラインのアイドル期間を総計し、現行の計画と比較したところ最大で 49%まで削減したことを確認した。さらに中組組立工程にて問題となっていたライン間の同期制約についても、先行中組ブロックが完成したのちに中組ブロックを組み立て開始する計画が作成できることを確認した。以上より Heap 法が制約条件を満たしつつ工程計画を行えることを確認することができた。また、工数平準化とアイドル期間削減を両立した工程計画を得るために、組合せ最適化問題として焼きなまし法を用いた製作順序の調査を行った。その結果、アイドル期間を最小にしようと試みた場合には人員稼働率が低下し、アイドル期間を許容すると人員稼働率が向上する傾向があることが明確となった。また人員稼働率を向上するためには、時数のかかる作業を分散させる製作順序が重要となることがわかったが、これは現実に人員平準化に対して行われる取り組みに即した結果であることがわかった。

造船所 B については、造船所全体を対象として適用を行った。ここでは組立工程計画の自動化を目的として、焼きなまし法による定盤計画自動決定を行った。これによりアイドル期間を半減させつつ、リード期間の 2 日の短縮を達成する工程計画例を得ることが出来た。また、造船所全体をリソースとして定義しモデル化を行った結果、生産計画効率化のボトルネックとなる設備が塗装工場であることを明確にした。これにより

Heap 法を用いることで工程計画改善のためにネックとなっている設備がどこか調査できることがわかった。

最後に Heap 法を実際の作業者が扱うことを考えて、Heap 法によるスケジューリングツールとして WEB アプリケーション化した。これによりプログラムの知識なく直感的な操作で Heap 法を用いることが可能となった。

本研究により新しい計画手法である Heap 法の提案と、その実用性の確認を行うことが出来た。今後の課題を以下に述べる。

適用する工程や作業内容によってリソースとピースの定義を行う必要がある。このピースを正しく定義しなくては作業間のつながりに不具合が起き、実行できない計画を作成することに繋がる。そのため、ピースの定義をより単純な操作で行える仕組みを開発する必要がある。

造船所の実データに適用した結果、アイドル期間の削減やリード期間の短縮を図れることがわかった。しかしこの計画が実際に実行可能かどうかについて調査が進んでいない。そのためモデルとなった造船所に対して Heap 法によって得られたスケジューリング結果の精査を依頼して、計画の不備や改善点を調べる必要がある。

アイドル期間中は、ブロックをどこかに蔵置しておく必要があるが、工程計画後にアイドル期間が決定するために蔵置する区画が未定となっている。本研究であつかった造船所 B の場合、蔵置に台船を用いているために、蔵置区画が足りない場合に台船の追加を行うことで対応できる。しかし造船所 A の場合、ストックヤードが有限であるためアイドル期間中の蔵置場所を確保する必要がある。このように未定の状況を想定した工程計画が行えるように改良する必要がある。

本研究では、ある一定期間の計画を作成することを行った。実際の工程計画では、これまで立てた計画と次の期間の計画を繋げる作業が行われる。プル型スケジューリングによって、計画の納期側に設備的余裕がないため、期間の重なる部分の微調整が煩雑になることが予想される。このような工程計画間のオーバーラップを機械的に解決することが求められる。

最後に本研究に必要なブロック組立計画データを提供して頂いた株式会社大島造船所および株式会社臼杵造船所に心より感謝致します。

参考文献

[1] 岩下寛弥，梶原宏之：Heap モデルに基づくツインタンデム建造方式の組立日程計画，日本船舶海洋工学会論文集，第 21 号，pp.83-91 （2015）

[2] Hiroyuki Kajiwara, Tomoya Iwashita: Heap-model-based Scheduling of Ship Building Lines for Tandem Construction of a Pair of Ships, Proc. of 14th International Conference on Computer and IT Applications in the Maritime Industries, pp.208-222 (2015)

[3] 奥本泰久：造船技術と生産システム，成山堂書店（2009）

[4] 奥本泰久他：造船工作法，成山堂書店（2012）

[5] 山崎眞喜：造船の計画管理，成山堂書店（1995）

[6] 青山和浩，野本敏治，渡辺健太郎： ペトリネットを利用した造船工場シミュレータ，日本造船学会論文集，第 182 号， pp.795-807 （1997）

[7] 伊庭斉志：システム工学の基礎，システムのモデル化と制御，数理工学社（2007）

[8] 梶原宏之，中尾洋一：離散事象システム理論に基づく造船ラインのスケジューリング，日本船舶海洋工学会論文集，第 4 号，pp.89-94 （2006）

[9] 梶原宏之，人位康弘，中尾洋一，岩下寛弥：Max-Plus 代数に基づくツインタンデム建造方式の組立日程計画，日本船舶海洋工学会論文集，第 20 号，pp.205-220 (2014)

[10] B.Heidergott, G.J.Olsder, J.van der Woude: Max Plus at Work, Princeton University Press (2006)

[11] Hiroyuki Kajiwara, Khairul Hassan: Heap-model Based Approach to Pull-type Scheduling in Block Assembly Lines，日本船舶海洋工学会春季講演論文集 （2012）

[12] 電気学会編：進化技術ハンドブック，第 I 巻基礎編，近代科学社，pp.185-188 (2010)

[13] 岩下寛弥，梶原宏之，山村 仁，人位康弘：人員平準化を考慮したブロック組立の工程計画，日本船舶海洋工学会春季講演会論文集，pp.193-196 (2015)

[14] 岩下寛弥，梶原宏之：Heap 法に基づく日程計画用 WEB アプリケーションの開発，日本船舶海洋工学会春季講演会論文集，pp.389-392 (2016)