愛知淑徳大学論集―人間情報学部篇　第２号　2012 年３月，pp. 17-31

原著

卒業プロジェクト配属ソフトウェアの開発と最適化問題

Software Development and Optimization Problem in a System for Assigning Students to Laboratories

佐々木　　尚*

Takeshi SASAKI

要　旨

現在、愛知淑徳大学人間情報学部人間情報学科の２年生の学生数は 265 名である。彼らは来年度、14 ある卒業プロジェクトのいずれかに配属され、ゼミ活動を行う予定である。これら多数の学生について、卒業プロジェクトへの配属振り分け作業を行うことは容易ではない。できる限り学生の希望を通しつつ、振り分け作業の煩わしさを軽減するための一つの方法として、ソフトウェアの開発が考えられるであろう。そこで、本稿では、今回新しく開発した卒業プロジェクト配属ソフトウェアについて解説し、その結果について報告および議論する。

キーワード：ゼミ　配属　振り分け　最適化問題

１．背景

2010 年度、愛知淑徳大学に人間情報学部人間情報学科が新設され、本年度で２年目となる。来年度からは、新３年生が各教員の受け持つ卒業プロジェクトに配属され、各卒業プロジェクトにおいて卒業研究や卒業制作等を行う予定である。そこで問題になってくることの一つが、学生の卒業プロジェクトへの配属方法である。当然、学生は、将来就きたい仕事や研究したい内容に合わせて卒業プロジェクトを選択するであろう。その希望が叶わなかったときの絶望感を想像すると、教員側からの要求にばかり目を向けるわけにはいかない。学生にとって卒業プロジェクトへの配属は、とても大きな問題である。できるだけ学生の希望を叶えてあげられるように、各卒業プロジェクトへの配属を考えてあげなければならない。そのようにするためにはどのような戦略がよいであろうか？

これまで一般的に行われてきた方法としては、第一希望、第二希望、第三希望を書いた紙を学生に提出させ、それらをもとに教員側が振り分ける、というものであった。しかし、この手法が有効であるケースは、学科の学生数が少なく、且つ、事前に学生同士が情報交換してすり合わせを行っており、どこに希望を出せば通りやすいか、ということを熟知しているときである。しかしながら、現在本学科の学生数は 265 名と多く、14 もある卒業プロジェクトのそれぞれの個性も強いため、学生同士の事前のすり合わせに頼るわけにはいかない。これまでの手法とは全く違った手法による、卒業プロジェクト配属方法が必要である。またそれは、学生や教員の皆が納得できる配属方法でなければならない。これらを実現するためには、そもそも学生らが、それぞれの

*　愛知淑徳大学人間情報学部　tnsasaki @ asu.aasa.ac.jp

卒業プロジェクトに対してどのような評価やイメージを抱き、どれくらいその卒業プロジェクトに配属されたいのか、という情報を知らなければならない。これらを知ることは簡単ではないが、一つの方法として、各学生に各卒業プロジェクトへの配属希望度として１点から 100 点までのスコアとして表現してもらうことが考えられる。全ての学生について、それぞれの卒業プロジェクトへの配属希望度を表したスコアが分かれば、それをもとに学生の配属を考えることが可能となる。特に、これを最適化問題[1-3] の一つとして捉えることにより、この問題を解決することはそれほど難しいことではないように思える。

今回、この卒業プロジェクト配属の問題を一つの最適化問題として捉え、実際に C 言語によるソフトウェア開発を行い、その結果を卒業プロジェクト配属時に利用した。本稿では、今回作成したソースコードの公開および、そのアルゴリズムについて解説し、またその結果について報告および議論する。

２．目的

卒業プロジェクト配属問題を一つの最適化問題として捉え、学生から提出された各卒業プロジェクトに対するスコア表（配属希望度を表す）をもとに、各学生の配属希望をできる限り叶えるソフトウェアの開発を行うこと。

３．方法

⑴　アルゴリズムについて

ここでは、具体的にどのようにして学生を卒業プロジェクトに配属するのかについて述べたい。

はじめに取り組まなければならない作業がスコア表の作成である。スコアとは、学生がどの卒業プロジェクトに配属されたいか、その希望度を表現した数値データである。第一希望を 100 点とし、それに対して第二希望は何点、第三希望は何点、というようにして第 14 希望までそれぞれの卒業プロジェクトに対して配属希望度を数値化してもらう。最終的に、各学生がそれぞれの卒業プロジェクトに対して配属希望度を表したスコア表が得られる。この数値を規格化し、その新しいスコア（以下、規格化スコア）表を用いて卒業プロジェクト配属が決定される。規格化の詳細については、「⑶　ソースコードについて」内の「normalize」関数についての説明を参照のこと。

ある卒業プロジェクトに対して、第一希望者が 20 名以下の場合、その第一希望を出した学生は、このソフトウェアで振り分けられることなく配属が決定されている。また、第一希望者が 20 名を超えている場合も、教員側からの課題やレポート等によって篩にかけられて 20 名に絞られ、これらの学生は配属決定済みである。本ソフトウェアは、これらの事前作業が終了した時点で配属が決まっていない学生に対して実行される。今回の場合は学科の学生 265 名中 83 名が第一希望で配属できなかったため、この 83 名を本ソフトウェアにより配属決定した。つまり、これらの第一希望が通らなかった学生について、第二希望以下への振り分けを行ったということになる。

これらの学生に対して、どのように卒業プロジェクトへ振り分けていくかであるが、いくつかの卒業プロジェクトにまだ空きがある場合、当然そこにこれらの学生を配属していく、という戦略が考えられる。第二希望の通る学生は定員上限まで配属し、それに漏れた学生は第三希望の卒業プロジェクトへ配属可能かどうか検討し、それが無理ならば第四希望を検討し、……というようにするのである。ある卒業プロジェクトにまだ空きがあり、且つ、第二希望者が空き数よりも多数の場合、規格化スコアを用いて学生の配属優先順位を決める。つまり、第一希望ほどではないにしろ、第二希望の卒業プロジェクトに対して、そこにも強く配属希望をしている学生から優先的に配属してゆくのである。この時点で第二希望に配属されなかった学生は、第三希望、そしてそれがだめなら第四希望、……と回されてゆくことになる。

⑵ スコア表の取得について

学生の配属希望度を反映させたスコア表の作成についてであるが、これは学生に直接アンケートをとる必要がある。今回に関しては、教務委員会が準備したCGIのソフトウェアを利用して学生に提出させた。アンケート結果の回収方法はどのような手法を用いてもかまわないが、付録1のファイルscore.txtを準備する必要がある。ここでscore.txtファイルは、今回の人間情報学科でのアンケート結果をまとめたものである。紙面の都合により、卒業プロジェクト13、14のデータは削除した。ファイルscore.txtの1行目はDecision、S_ID、NAME、P_ID、P01、P02…P12（実際にはP14まで）となっている。以下に、これらの値の説明をする。

<Decision>

Decisionの列には0、1の数字が記述されている。ある卒業プロジェクトに対して、第一希望者が20名以下の場合、その第一希望を出した学生は、このソフトウェアで振り分けられることなく配属が決定されている。また、第一希望者が20名を超えている場合も、教員側からの課題やレポート等によって篩にかけられて20名に絞られ、これらの学生は配属決定済みである。既に配属が決定されている学生には1を、それ以外の学生は0を割り当てた。この0が割り当てられた学生に対して、本ソフトウェアによる配属が決定される。

<S_ID>

これは学生に割り当てられたIDである。プログラミング上、特に重要ではないが、今後の本ソフトウェアの機能拡張も考えて導入した。

<NAME>

これは文字通り学生の名前である。今回の付録1には学生名を伏せるため、学生A、学生B、などとした。

<P_ID>

卒業プロジェクトは全部で14ある。それらにIDが1から14まで割り当てられている。P_IDは、学生が第一希望とした卒業プロジェクトID。

<P01 ～ P12（実際にはP01 ～ P14）>

それぞれの卒業プロジェクトに対して、配属希望度を表したスコア。スコア値100は、その卒業プロジェクトに対して、その学生が第一希望であることを意味する。逆に、値の小さなスコアは、その卒業プロジェクトへの配属希望度の低さを意味する。

本ソフトウェアを用いる際には、付録1のscore.txtのようなファイルフォーマットで全ての学生のスコアデータを準備する必要がある。ちなみに、各データ間は“タブ区切り”を用いているため、その点に注意して作成する必要がある。もちろん“カンマ区切り”等を用いたい場合は、ソースコードを変更すればよい。

⑶ ソースコードについて

スコア表が完成したら、本ソフトウェアの実行に移る。本ソフトウェアの役割は、第一希望が通らなかった学生たちの第二希望以降への振り分けである。従って、既に第一希望で配属が決定された学生はDecisionの値が1となっており、本ソフトウェアでは無視される。付録2（sort.c）を参照していただきたい。

●パラメータについて

<N_proj>

卒業プロジェクトの数。

<MEM_num>

全学生数。第一希望が通り、配属が既に決まっている学生も含む。

<PROJ_mem_max>

本ソフトウェアによって各卒業プロジェクトに振り分けられる学生の最大数。ただし、事前に用意されたスコア表の Decision を読み取ったときに、ある卒業プロジェクトの学生数が PROJ_mem_max 値を超えることは許されている。今回の人間情報学部人間情報学科に関わる配属の場合、PROJ_mem_max＝ 19 としたため、５つの卒業プロジェクトの配属学生数が 20 名となり、PROJ_mem_max 値を超えた。PROJ_mem_max の値はいくつか試して、最適な数値を探すとよい。

●関数について

<dataread>

スコアデータを読み込む関数。

<calc_ranking>

それぞれの学生について、各卒業プロジェクトに対する希望順位を計算する関数。

<calc_fix>

第一希望が通らなかった学生を、規格化スコアに応じて第二希望から順々に配属していく関数。配属先は、配属学生数が PROJ_mem_max を超えていない卒業プロジェクトに対して学生を配属していく。配属学生数が PROJ_mem_max になった卒業プロジェクトに対しては、学生の配属をそこでやめる。第二希望が通らなかった学生は第三希望へと回り、それがだめなら第四、第五、……となってゆく。

<student_rank_sort>

卒業プロジェクト X に対して、第 Y 希望とした学生間の規格化スコアによるランク付けを行う関数。ランキング上位の学生は希望が通りやすい。calc_fix 内で利用されている。

<calc_i_rank_counter>

第一希望が通らなかった学生について、卒業プロジェクト X に第 Y 希望の学生が何人いるかをカウントする関数。calc_fix 内で利用されている。

<normalize>

スコア表のデータをもとに、各学生に対してスコアの規格化を行う関数。分母はスコアの合計、分子は各卒業プロジェクトへの配属希望度を表したスコアであり、１未満の小数で表現されている。main で呼び出され、その結果が student_rank_sort 内で利用されている。

●構造体 Student のメンバについて

<fix_flag>

０のときは第一希望が通らなかった学生を意味し、１のときはその学生が第一希望で配属決定済みであることを意味する。Decision を読み込んでいる。本ソフトウェアによって配属が決定される学生は、この値が０の学生ということになる。

<proj>

初期値として、各学生がどの卒業プロジェクトに配属されたかを表す。本ソフトウェアで、この値がはじめ０に設定される学生は、卒業プロジェクト配属が決まっていないということであり、fix_flag＝ 0。本ソフトウェアで振り分けられる学生。

<id>

学生の ID。

<name>

学生の名前。

<score>

学生の卒業プロジェクトへの配属希望度を表すスコア。

<fscore>
score の値を規格化したもの。normalize 内で生成される。
<choice_ranking>
学生毎に、スコアを用いてそれぞれの卒業プロジェクトを第二希望から第 N_proj 希望までランク付けしたもの。

●その他の配列について
<iProj_counter>
それぞれの卒業プロジェクトに配属されている学生数をカウントし、その値を格納する配列。
<i_rank_counter>
第一希望が通らなかった学生について、卒業プロジェクト X に第 Y 希望の学生が何人いるかをカウントし、その結果を格納する配列。calc_fix 内で利用されている。
<i_Student_rank>
卒業プロジェクト X、第 Y 希望に関する学生間の規格化スコアによるランク付けを行った結果を格納する配列。ランキング上位の学生は希望が通りやすい。calc_fix 内で利用されている。

4．卒業プロジェクト配属の結果

付録 3 にこのソフトウェアの実行結果（output.txt）を付した。その出力結果について説明する。まず出力結果の言葉の意味については以下を参照。
<proj_id>
卒業プロジェクトの番号
< 配属人数 >
最終的に配属された学生数
<stu_id>
学生の ID
<flag>
配属が完了している学生はこの値が 1 になっている。ソフトウェア実行後に、この値が 0 となっている学生がいる場合は、何らかの問題が発生している可能性があり、ソースコードのチェックが必要。

出力結果の 1 行目の「proj_id = 0　配属人数 83」が意味するものは、第一希望が通らずに、本ソフトウェアによって卒業プロジェクトに配属された学生数を表す。proj_id = 1 から proj_id = 14 までの行は、本ソフトウェアを実行した結果、それぞれの卒業プロジェクトに配属された人数を表す。

次に続く stu_id を含む行は、学生 X が、どの卒業プロジェクトに、第 Y 希望で配属されたかを示している。参考のため、stu_id = 0 から 13 までと、stu_id = 256 から 264 までのデータを掲載した。データが多いため、波線は省略したデータを表す（以下同様）。そして stu_id の行を過ぎると、再び proj_id の行が出現する。これは、各卒業プロジェクトに、学生 X が、第 Y 希望で配属されていることを示す。参考として、卒業プロジェクト 6 と 14 のデータを掲載した。

以上のように、ソースコード上で出力方法に変更を加えれば、いくらでもほしいデータを取り出すことが可能である。例えば、配属結果から、第 X 希望の叶えられた学生が何名いたか、という結果も取得可能であり、表 1 のようであった。第一希望の学生 182 名は希望通り配属され、それ以外の 83 名が本ソフトウェアによって配属された。大まかな傾向として、第二希望から第十四希望にかけて、徐々に学生数が減少していることが

見てとれる。

５．考察

本ソフトウェアでは、第一希望の卒業プロジェクトへの配属が叶わなかった学生に対して、規格化スコアをもとに、第二希望以降の卒業プロジェクトへの配属を行った。結果としては、表１にあるように、全体的な傾向としては、第二希望から第十四希望にかけて、徐々に学生数が減少するように配属することができた。しかしながら、第八希望以降の卒業プロジェクトへの配属が行われた学生もおり、改善すべき点があるように思える。おそらく、今回一つ問題を挙げるとすれば、第一希望、第二希望、第三希望……という枠に当てはめて考えている点であろう。なぜならば、第一希望、第二希望、第三希望の卒業プロジェクトへの配属希望度であるスコアが同程度である学生、また第二希望から第六希望までのスコアがほぼ同じ、という学生もいるだろうことは容易に想像がつくからである。にもかかわらず、それらスコアの微差に注目し、第二希望から順に配属してゆく、という戦略をとってしまったのでは、表１のような結果になってもおかしくない。ただ、教務的な要求や時間的制約等により、今回はこのような戦略を取らざるを得なかった。時間的余裕があり、自由な発想で取り組んでいいのであれば、次のような戦略も考えられる。まず今回得られた結果を最終的な卒業プロジェクト配属結果とするのではなく、これを初期値として用いることを考える。つまり、状況としては表１をスタート地点とするのである。この状況から、さらに学生を卒業プロジェクト間でシャッフルすることを考える。第何希望、という枠で考えるのではなく、純粋に規格化スコアをもとに、学生を卒業プロジェクト間で交換してゆくのである。例えば、卒業プロジェクトＪに配属された学生Ａが卒業プロジェクトＫに移るとき、それほど規格化スコアの値が下がらないとする。このとき、卒業プロジェクトＫに配属された学生が卒業プロジェクトＪに移ったほうが、規格化スコアの値の上昇が著しいとすれば、この学生たちを交換したほうがよい。このような作業にメトロポリス法[4]などを用いて時間をできるだけかけて実行すれば、さらに学生の要求を満たした結果が得られると思われる。評価関数には、学生 265 名の配属先における規格化スコアの総和を用いて行えばよい。この総和が高ければ高いほど、学生の満足度も高いということになる。

表１

第一希望	182
第二希望	21
第三希望	17
第四希望	13
第五希望	10
第六希望	1
第七希望	1
第八希望	4
第九希望	6
第十希望	3
第十一希望	4
第十二希望	2
第十三希望	1
第十四希望	0
計	265

このように、本ソフトウェアは未だ発展途上ではあるが、ソフトウェア実行時間としては１秒程度であるため非常に有用でもある。今後の展望としては、上述した改良案を取り込みつつ、実行時間の短くて済むソフトウェアの開発を行いたいと考えている。

参考文献

［１］J. H. Holland, Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence, University of Michigan Press (1975)

［２］S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by simulated annealing”, *Science* **220** pp. 671–680 (1983)

［３］Y. Okamoto, Generalized-ensemble algorithms : enhanced sampling techniques for Monte Carlo and molecular dynamics simulations, *J. Mol. Graph. Model* **22**(5) pp. 425–439 (2004)

［４］N. Metropolis, et. al. : Equations of state calculations by fast computing machines, *J. Chem. Phys.* **21** pp. 1087–1091 (1953)

付録１（score.txt）

Decision	S_ID	NAME	P_ID	P01	P02	P03	P04	P05	P06	P07	P08	P09	P10	P11	P12
0	10001	学生 A	8	38	83	2	70	99	3	42	100	20	24	90	19
1	10002	学生 B	2	42	100	1	70	90	10	20	75	60	96	65	88
0	10003	学生 C	5	97	80	81	99	100	79	96	98	95	77	94	76
0	10004	学生 D	4	25	95	50	100	97	94	15	96	91	93	20	99
1	10005	学生 E	4	7	2	1	100	99	5	6	98	8	3	97	96
0	10006	学生 F	4	3	12	8	100	2	4	10	9	99	1	11	6
0	10007	学生 G	4	1	95	25	100	12	60	2	7	74	4	3	38
1	10008	学生 H	2	1	100	30	20	10	95	3	25	15	7	3	75
1	10009	学生 I	5	40	20	4	6	100	69	5	15	99	2	7	39
1	10010	学生 J	9	1	2	6	60	10	12	11	20	100	8	15	3
1	10011	学生 K	13	10	65	60	96	99	20	30	97	55	98	50	95
1	10012	学生 L	10	15	20	16	73	85	45	69	74	72	100	42	71
1	10013	学生 M	5	70	4	2	90	100	5	80	50	95	1	75	7
1	10014	学生 N	11	50	2	1	16	19	4	30	17	20	18	100	5
0	10015	学生 O	5	32	48	23	46	100	72	52	85	43	27	39	24
0	10016	学生 P	5	3	5	1	40	100	2	4	7	9	8	6	30
1	10017	学生 Q	14	12	45	35	80	20	13	11	10	25	9	14	75
1	10018	学生 R	12	70	90	30	99	95	80	60	94	98	97	50	100
0	10019	学生 S	8	9	30	4	8	7	6	95	100	99	1	20	3
1	10020	学生 T	4	80	4	1	100	75	3	85	70	8	7	90	6
1	10021	学生 U	8	35	22	15	85	80	10	90	100	65	28	40	25

付録２（sort.c）

```
#include<stdio.h>
#include<math.h>
#include<string.h>
#include<stdlib.h>
#include<time.h>

// プロジェクト数
#define N_proj 14
// 学生数
#define MEM_num 265

// プロジェクトメンバ数最大値（ただし、既に 19 名を超える数の
// 学生の配属が決定しているプロジェクトは、この限りではない）
#define PROJ_mem_max 19

// スコアデータ読み込み関数
void dataread(int i, char *buff);
// 学生毎に各プロジェクトの希望順位を決める関数
void calc_ranking(void);
// 第二希望から順に学生をプロジェクトに配属する関数
void calc_fix(void);
// プロジェクト j,第 i_rank 希望,に関する学生間の規格化スコアによるランク付けする関数
void student_rank_sort(int j, int i_rank);
// 第 1 希望が通らなかった学生について、
// あるプロジェクトに第 X 希望の学生が何人いるかをカウントする関数
void calc_i_rank_counter(void);
// スコアの規格化を行う関数
```

```
void normalize(void);

FILE *score_open;
FILE *output_open;


char buff[1000];

// 学生に関するデータの構造体
typedef struct{
    //0 のときは第一希望が通らなかった学生。のときは第一希望で配属決定。
    // つまり、の学生は本ソフトウェアでは取扱わない。
    int fix_flag[MEM_num];
    // 初期値として、学生がどのプロジェクトに配属されたかを表わす。
    //0 というプロジェクト配属に関する値は第一希望が通らなかった学生を意味する。
    int proj[MEM_num];
    // 学生の ID
    int id[MEM_num];
      // 学生の名前
      char name[MEM_num][100];
    // 学生のプロジェクトに関するスコアデータ
    // 第一希望のプロジェクトは点を与えた
    int score[MEM_num][N_proj+1];
      // 規格化後のデータ
      float fscore[MEM_num][N_proj+1];
    // スコアデータを使って、第二希望から第 N_proj 希望までランク付けしたもの。
    // ある学生について、X プロジェクトは第何希望か、を表わす。
    int choice_ranking[MEM_num][N_proj+1];
}PS;

PS Student;

// 初期値(第一希望)において、プロジェクトに既に配属されている学生の数をカウント
int iProj_counter[N_proj+1];

// 第 1 希望が通らなかった学生について、
// あるプロジェクトに第 X 希望の学生が何人いるかをカウント
int i_rank_counter[N_proj+1][N_proj+1];

// 規格化スコアで評価したときに,ある学生が,あるプロジェクトの第？希望に何番目に振り分けられるかを表わす。
int i_Student_rank[MEM_num][N_proj+1][N_proj+1];

int main(void)
{
    int i,j,sum_1=0;

    // データの読み込み
      score_open=fopen("selected_ver2_project2011.txt","r");
    if(score_open==NULL){
      perror("failed open score_file");
      exit(1);
    }

//      最初の 1 行分は読み捨てる↓
      fgets(buff,1000,score_open);
```

```
    for(i=0;fgets(buff,1000,score_open)!=NULL;i++){
            if(i==1)printf(buff);
       dataread(i,buff);
    }

       fclose(score_open);

       // スコアの規格化
       normalize();

    // 既にプロジェクトに配属されている学生の数を数える。
    for(i=0;i<=N_proj;i++){
       iProj_counter[i]=0;
    }
    for(i=0;i<MEM_num;i++){
            if(Student.fix_flag[i]==1){
     iProj_counter[Student.proj[i]]=iProj_counter[Student.proj[i]]+1;
            }else{
                    // まだプロジェクトに配属されていない学生
                    Student.proj[i]=0;
                    iProj_counter[0]=iProj_counter[0]+1;
            }
    }
       for(i=0;i<=N_proj;i++){
            printf("i=%d  proj=%d¥n",i,iProj_counter[i]);
            sum_1=sum_1+iProj_counter[i];
            printf("sum=%d¥n",sum_1);
    }

    //choice_ranking の値を決める。
    // スコアが同点の場合は、プロジェクト番号の大きいほうが優先される
    calc_ranking();
    for(i=0;i<MEM_num;i++){
       for(j=1;j<=N_proj;j++){
    printf("i=%d,j=%d,ranking=%d¥n",i,j,Student.choice_ranking[i][j]);
       }
    }

       calc_fix();

       for(j=0;j<=N_proj;j++){
            printf("%d  project  member=%d¥n",j,iProj_counter[j]);
       }

        output_open=fopen("output.txt","w");
        if(output_open==NULL){
           perror("failed  open  output_file");
           exit(1);
        }
       for(j=0;j<=N_proj;j++){
fprintf(output_open,"proj_id=%d¥t 配属人数＝ %d¥n",j,iProj_counter[j]);
       }
       for(i=0;i<MEM_num;i++){
  fprintf(output_open,"stu_id=%d¥t%8s¥tflag=%d¥tproj_id=%d¥t 第 %d 希望 ¥n",i,Student.name[i],Student.fix_flag[i],
```

```
Student.proj[i],Student.choice_ranking[i][Student.proj[i]]);
    }
        for(j=1;j<=N_proj;j++){
                for(i=0;i<MEM_num;i++){
                        if(Student.proj[i]==j){
                                fprintf(output_open,"proj_id=%d¥t%8s¥t 第 %d 希望 ¥n",j,Student.name[i],Student.choice_
ranking[i][Student.proj[i]]);
                        }
                }
        }
        fclose(output_open);

// 各プロジェクトのスコアに同点を付けてしまっている学生
        /*
        int j2;

        for(i=0;i<MEM_num;i++){
                for(j=1;j<=N_proj-1;j++){
                        for(j2=j+1;j2<=N_proj;j2++){

        if(Student.fscore[i][j]==Student.fscore[i][j2]){
                                        printf("doutenn i=%d   j=%d j2=%d %s¥n",i,j,j2,Student.name[i]);

                                }
                        }
                }
        }
    */

        return 0;
}

void calc_fix(void)
{
    int i,j,i_rank,sum=0;
        int i_counter=0;

        calc_i_rank_counter();

    // 第二希望から順に詰めていく
    for(i_rank=2;i_rank<=N_proj;i_rank++){
                calc_i_rank_counter();
        for(j=1;j<=N_proj;j++){
                if(iProj_counter[j]<PROJ_mem_max){
    // あるプロジェクトが定員上限になっていなければ学生を詰めこむ
if(iProj_counter[j]+i_rank_counter[j][i_rank]<=PROJ_mem_max){
iProj_counter[j]=iProj_counter[j]+i_rank_counter[j][i_rank];
for(i=0;i<MEM_num;i++){
if(Student.fix_flag[i]==0&&Student.choice_ranking[i][j]==i_rank){
        Student.proj[i]=j;
        Student.fix_flag[i]=1;
        i_counter=i_counter+1;
        printf("i=%d  Student.proj=%d
        Student.fix_flag=%d¥n",i,Student.proj[i],Student.fix_flag[i]);
                                        printf("i=%d  j=%d
```

```
        Student.choice_ranking=%d i_rank=%d¥n",i,j,Student.choice_ranking[i][j],i_rank);
                                }
                        }
                        // 定員オーバーになった場合
                }else
        if(iProj_counter[j]+i_rank_counter[j][i_rank]>PROJ_mem_max){
// プロジェクト j,第 i_rank 希望,に関する学生間の規格化スコアによるランク付け
        student_rank_sort(j,i_rank);
        for(i=0;i<MEM_num;i++){

if(Student.fix_flag[i]==0&&Student.choice_ranking[i][j]==i_rank){
if(i_Student_rank[i][j][i_rank]<=(PROJ_mem_max-iProj_counter[j])){
                                        Student.proj[i]=j;
                                        Student.fix_flag[i]=1;
                                        i_counter=i_counter+1;
                                }
                        }
                        }
                        iProj_counter[j]=PROJ_mem_max;
                }
                  }
        }
    }

for(i=0;i<MEM_num;i++){
        printf("i=%d flag=%d proj=%d kibou_rank=%d ¥n",i,Student.fix_flag[i],Student.proj[i],Student.choice_ranking[i]
[Student.proj[i]]);
    }

        printf("i_counter=%d¥n",i_counter);
}


void student_rank_sort(int j, int i_rank)
{

    int i,i2;

    // 初期化
    for(i=0;i<MEM_num;i++){
    if(Student.fix_flag[i]==0&&Student.choice_ranking[i][j]==i_rank){
            i_Student_rank[i][j][i_rank]=1;
        }
    }

  for(i=0;i<MEM_num-1;i++){
  if(Student.fix_flag[i]==0&&Student.choice_ranking[i][j]==i_rank){
  for(i2=i+1;i2<MEM_num;i2++){
  if(Student.fix_flag[i2]==0&&Student.choice_ranking[i2][j]==i_rank){
        if(Student.fscore[i][j]>=Student.fscore[i2][j]){

        i_Student_rank[i2][j][i_rank]=i_Student_rank[i2][j][i_rank]+1;
        }else if(Student.fscore[i][j]<Student.fscore[i2][j]){
        i_Student_rank[i][j][i_rank]=i_Student_rank[i][j][i_rank]+1;
                }
```

```
                }
            }
        }
    }
}

void calc_ranking(void)
{
    int i,j,k,l;

    // まずは初期化。
    for(i=0;i<MEM_num;i++){
        for(j=1;j<=N_proj;j++){
            Student.choice_ranking[i][j]=N_proj;
        }
    }

    // 学生毎に各プロジェクトの希望順位を決める
    for(i=0;i<MEM_num;i++){
        for(j=1;j<=N_proj-1;j++){
            for(k=j+1;k<=N_proj;k++){
                if(Student.score[i][j]<=Student.score[i][k]){
          Student.choice_ranking[i][k]=Student.choice_ranking[i][k]-1;
                }else if(Student.score[i][j]>Student.score[i][k]){
          Student.choice_ranking[i][j]=Student.choice_ranking[i][j]-1;
                }
            }
        }
    }
}

void dataread(int i, char *buff)
{

        sscanf(buff,"%d¥t%d¥t%s¥t%d¥t%d¥t%d¥t%d%d¥t%d¥t%d%d¥t%d¥t%d%d¥t%d¥t%d%d¥t%d"
,&Student.fix_flag[i],&Student.id[i],Student.name[i],&Student.proj[i],
        &Student.score[i][1],
        &Student.score[i][2],
        &Student.score[i][3],
        &Student.score[i][4],
        &Student.score[i][5],
        &Student.score[i][6],
        &Student.score[i][7],
        &Student.score[i][8],
        &Student.score[i][9],
        &Student.score[i][10],
        &Student.score[i][11],
        &Student.score[i][12],
        &Student.score[i][13],
        &Student.score[i][14]);

}

void calc_i_rank_counter(void)
{
```

```
        int i,j,i_rank;

        // 初期化
        for(j=1;j<=N_proj;j++){
        for(i_rank=2;i_rank<=N_proj;i_rank++){
             i_rank_counter[j][i_rank]=0;
        }
      }

// 第一希望が通らなかった学生について、
      // あるプロジェクトに第 X 希望の学生が何人いるかをカウント
      for(i=0;i<MEM_num;i++){
        if(Student.fix_flag[i]==0){
             for(j=1;j<=N_proj;j++){
                for(i_rank=2;i_rank<=N_proj;i_rank++){
                     if(Student.choice_ranking[i][j]==i_rank){

        i_rank_counter[j][i_rank]=i_rank_counter[j][i_rank]+1;
                     }
                }
             }
        }
      }

}

void normalize(void)
{

        int i,j;
        int sum[MEM_num];

        // 初期化
        for(i=0;i<MEM_num;i++){
                sum[i]=0;
        }

        for(i=0;i<MEM_num;i++){
                for(j=1;j<=N_proj;j++){
                        sum[i]=sum[i]+Student.score[i][j];
                }
        }

        for(i=0;i<MEM_num;i++){
                for(j=1;j<=N_proj;j++){
Student.fscore[i][j]=(float)(Student.score[i][j])/(float)(sum[i]);
                        printf("fscore=%f¥n",Student.fscore[i][j]);
                        printf("i=%d    sum=%d¥n",i,sum[i]);
                }
        }
}
```

付録３（output.txt）

```
proj_id＝0	配属人数=83
proj_id＝1	配属人数=18
proj_id＝2	配属人数=20
proj_id＝3	配属人数=16
proj_id＝4	配属人数=20
proj_id＝5	配属人数=20
proj_id＝6	配属人数=19
proj_id＝7	配属人数=18
proj_id＝8	配属人数=20
proj_id＝9	配属人数=19
proj_id＝10	配属人数=19
proj_id＝11	配属人数=18
proj_id＝12	配属人数=20
proj_id＝13	配属人数=19
proj_id＝14	配属人数=19
stu_id＝0	学生 A	flag＝1	proj_id＝1	第８希望
stu_id＝1	学生 B	flag＝1	proj_id＝2	第１希望
stu_id＝2	学生 C	flag＝1	proj_id＝1	第４希望
stu_id＝3	学生 D	flag＝1	proj_id＝13	第３希望
stu_id＝4	学生 E	flag＝1	proj_id＝4	第１希望
stu_id＝5	学生 F	flag＝1	proj_id＝9	第２希望
stu_id＝6	学生 G	flag＝1	proj_id＝13	第３希望
stu_id＝7	学生 H	flag＝1	proj_id＝2	第１希望
stu_id＝8	学生 I	flag＝1	proj_id＝5	第１希望
stu_id＝9	学生 J	flag＝1	proj_id＝9	第１希望
stu_id＝10	学生 K	flag＝1	proj_id＝13	第１希望
stu_id＝11	学生 L	flag＝1	proj_id＝10	第１希望
stu_id＝12	学生 M	flag＝1	proj_id＝5	第１希望
stu_id＝13	学生 N	flag＝1	proj_id＝11	第１希望
～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～
～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～
stu_id＝256	学生 O	flag＝1	proj_id＝8	第１希望
stu_id＝257	学生 P	flag＝1	proj_id＝5	第１希望
stu_id＝258	学生 Q	flag＝1	proj_id＝10	第１希望
stu_id＝259	学生 R	flag＝1	proj_id＝12	第１希望
stu_id＝260	学生 S	flag＝1	proj_id＝5	第１希望
stu_id＝261	学生 T	flag＝1	proj_id＝10	第２希望
stu_id＝262	学生 U	flag＝1	proj_id＝9	第１希望
stu_id＝263	学生 V	flag＝1	proj_id＝12	第１希望
stu_id＝264	学生 W	flag＝1	proj_id＝12	第１希望
～～～～～～～～～～～～～～～～～～～～～～
～～～～～～～～～～～～～～～～～～～～～～
proj_id＝6	学生 A	第２希望
proj_id＝6	学生 B	第３希望
proj_id＝6	学生 C	第１希望
proj_id＝6	学生 D	第１希望
proj_id＝6	学生 E	第１希望
proj_id＝6	学生 F	第３希望
proj_id＝6	学生 G	第１希望
proj_id＝6	学生 H	第１希望
proj_id＝6	学生 I	第１希望
proj_id＝6	学生 J	第１希望
proj_id＝6	学生 K	第１希望
```

proj_id＝6　　学生 L　　第 1 希望
proj_id＝6　　学生 M　　第 1 希望
proj_id＝6　　学生 N　　第 1 希望
proj_id＝6　　学生 O　　第 1 希望
proj_id＝6　　学生 P　　第 1 希望
proj_id＝6　　学生 Q　　第 1 希望
proj_id＝6　　学生 R　　第 1 希望
proj_id＝6　　学生 S　　第 1 希望
～～～～～～～～～～～～～～～～～～～～～～
～～～～～～～～～～～～～～～～～～～～～～
proj＝14　　学生 A　　第 1 希望
proj＝14　　学生 B　　第 1 希望
proj＝14　　学生 C　　第 3 希望
proj＝14　　学生 D　　第 1 希望
proj＝14　　学生 E　　第 2 希望
proj＝14　　学生 F　　第 1 希望
proj＝14　　学生 G　　第 3 希望
proj＝14　　学生 H　　第 1 希望
proj＝14　　学生 I　　第 1 希望
proj＝14　　学生 J　　第 2 希望
proj＝14　　学生 K　　第 1 希望
proj＝14　　学生 L　　第 4 希望
proj＝14　　学生 M　　第 2 希望
proj＝14　　学生 N　　第 1 希望
proj＝14　　学生 O　　第 4 希望
proj＝14　　学生 P　　第 1 希望
proj＝14　　学生 Q　　第 1 希望
proj＝14　　学生 R　　第 3 希望
proj＝14　　学生 S　　第 1 希望