Job Shop

Generated by Doxygen 1.8.14

Contents

1 Data Structure Index **1**

1.1 Data Structures 1

2 File Index **3**

2.1 File List 3

3 Data Structure Documentation **5**

3.1 BLIST Struct Reference 5

3.1.1 Detailed Description 5

3.1.2 Field Documentation 5

3.1.2.1 machine 6

3.1.2.2 makespan 6

3.1.2.3 order 6

3.2 JOB Struct Reference 6

3.2.1 Detailed Description 7

3.2.2 Field Documentation 7

3.2.2.1 estime 7

3.2.2.2 magic 7

3.2.2.3 mhtime 7

3.2.2.4 next 7

3.2.2.5 order 8

3.2.2.6 prev 8

3.2.2.7 process_time 8

3.2.2.8 start 8

3.2.2.9 step . . . 8
3.3 JOBMACHINEPAR Struct Reference . . . 8
3.3.1 Detailed Description . . . 9
3.3.2 Field Documentation . . . 9
3.3.2.1 job . . . 9
3.3.2.2 machine . . . 9
3.4 MACHINEORDER Struct Reference . . . 9
3.4.1 Detailed Description . . . 10
3.4.2 Field Documentation . . . 10
3.4.2.1 machines . . . 10
3.4.2.2 size . . . 10
3.5 NEIGHBOR Struct Reference . . . 10
3.5.1 Detailed Description . . . 11
3.5.2 Field Documentation . . . 11
3.5.2.1 next . . . 11
3.5.2.2 prev . . . 11
3.6 ONEMACHINE_BRANCH_AND_BOUND_ASSISTANT Struct Reference . . . 11
3.6.1 Detailed Description . . . 12
3.6.2 Field Documentation . . . 12
3.6.2.1 active . . . 12
3.6.2.2 bound . . . 12
3.7 ONEMACHINestime Struct Reference . . . 12
3.7.1 Detailed Description . . . 13
3.7.2 Field Documentation . . . 13
3.7.2.1 estime . . . 13
3.7.2.2 mhtime . . . 13
3.7.2.3 process_time . . . 13
3.8 PAIR_ASSISTANT_TYPE Struct Reference . . . 14
3.8.1 Detailed Description . . . 14
3.8.2 Field Documentation . . . 14
3.8.2.1 job_num . . . 14
3.8.2.2 mach_num . . . 15
3.8.2.3 proc_time . . . 15
3.8.2.4 start_time . . . 15
3.8.2.5 step . . . 15
3.9 SEQUENCE Struct Reference . . . 15
3.9.1 Detailed Description . . . 16
3.9.2 Field Documentation . . . 16
3.9.2.1 job . . . 16

4 File Documentation **17**

4.1 bottle.c File Reference . . . 17

4.1.1 Detailed Description . . . 18

4.1.2 Macro Definition Documentation . . . 18

4.1.2.1 TRY_COUNT . . . 18

4.1.3 Typedef Documentation . . . 19

4.1.3.1 blist_t . . . 19

4.1.3.2 mo_t . . . 19

4.1.3.3 neighbor_t . . . 19

4.1.4 Function Documentation . . . 19

4.1.4.1 clear_and_backup_seq() . . . 19

4.1.4.2 clear_seq() . . . 20

4.1.4.3 cp_mo() . . . 20

4.1.4.4 cp_neighbor() . . . 21

4.1.4.5 cp_seq() . . . 21

4.1.4.6 critical() . . . 21

4.1.4.7 eval() . . . 22

4.1.4.8 re_optimization_phase_1() . . . 22

4.1.4.9 re_optimization_phase_2() . . . 23

4.1.4.10 restore_neighbor() . . . 24

4.1.4.11 save_times() . . . 25

4.1.4.12 set_seq() . . . 25

4.1.4.13 shifting_bottle_neck() . . . 25

4.1.5 Variable Documentation . . . 27

4.1.5.1 best_makespan . . . 27

4.2 bottle.h File Reference . . . 27

4.2.1 Detailed Description . . . 28

4.2.2 Macro Definition Documentation . . . 28

4.2.2.1 INFINITAS . . . 28

4.2.2.2 MAX . . . 28

4.2.2.3 MAXJOB . . . 28
4.2.2.4 MAXMACHINE . . . 29
4.2.3 Typedef Documentation . . . 29
4.2.3.1 job_t . . . 29
4.2.3.2 onemach_times_t . . . 29
4.2.3.3 sequence_t . . . 29
4.2.4 Function Documentation . . . 29
4.2.4.1 one_machine() . . . 29
4.2.4.2 prestissimo() . . . 30
4.2.4.3 run_bottle_neck() . . . 31
4.2.5 Variable Documentation . . . 32
4.2.5.1 job . . . 32
4.2.5.2 job_size . . . 32
4.2.5.3 machine_size . . . 33
4.2.5.4 terminate_flag . . . 33
4.3 common_definition.c File Reference . . . 33
4.3.1 Variable Documentation . . . 33
4.3.1.1 job . . . 33
4.3.1.2 job_size . . . 34
4.3.1.3 machine_size . . . 34
4.3.1.4 terminate_flag . . . 34
4.4 eval.c File Reference . . . 34
4.4.1 Detailed Description . . . 35
4.4.2 Typedef Documentation . . . 35
4.4.2.1 job_machine_t . . . 35
4.4.3 Function Documentation . . . 35
4.4.3.1 eval() . . . 35
4.4.4 Variable Documentation . . . 36
4.4.4.1 magicnum . . . 36
4.5 io.c File Reference . . . 36

4.5.1 Detailed Description . . . 37
4.5.2 Typedef Documentation . . . 37
4.5.2.1 pair_ass_t . . . 37
4.5.3 Function Documentation . . . 37
4.5.3.1 getprob() . . . 37
4.5.3.2 machine_sort_cmp() . . . 38
4.5.3.3 prestissimo() . . . 39
4.5.3.4 starttime_sort_cmp() . . . 39
4.5.3.5 write_file() . . . 40
4.5.4 Variable Documentation . . . 41
4.5.4.1 best_makespan . . . 41
4.6 main.c File Reference . . . 41
4.6.1 Detailed Description . . . 41
4.6.2 Function Documentation . . . 42
4.6.2.1 main() . . . 42
4.7 onemachine.c File Reference . . . 42
4.7.1 Detailed Description . . . 43
4.7.2 Macro Definition Documentation . . . 43
4.7.2.1 ONEMACH_BBNODES . . . 43
4.7.3 Typedef Documentation . . . 43
4.7.3.1 onemach_bb_ass_t . . . 44
4.7.4 Function Documentation . . . 44
4.7.4.1 cp_onemach_time() . . . 44
4.7.4.2 lower_bound() . . . 44
4.7.4.3 make_span() . . . 45
4.7.4.4 make_value() . . . 46
4.7.4.5 mwr_schedule() . . . 46
4.7.4.6 one_machine() . . . 46

Index **49**

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

BLIST 5
JOB 6
JOBMACHINEPAR 8
MACHINEORDER 9
NEIGHBOR 10
ONEMACHINE_BRANCH_AND_BOUND_ASSISTANT 11
ONEMACHINestime 12
PAIR_ASSISTANT_TYPE 14
SEQUENCE 15

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

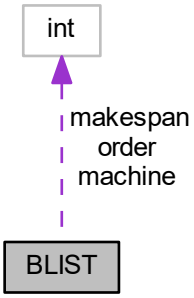
bottle.c
    Core algorithms 17
bottle.h
    Header file for the whole project 27
common_definition.c 33
eval.c
    Makespan 34
io.c
    IO 36
main.c
    Enterpoint 41
onemachine.c
    One-machine sequencing 42

Chapter 3

Data Structure Documentation

3.1 BLIST Struct Reference

Collaboration diagram for BLIST:



Data Fields

- int machine
- int makespan
- int order [MAXJOB]

3.1.1 Detailed Description

Store the bottle informnation.

3.1.2 Field Documentation

3.1.2.1 machine

```
int machine
```

Machine number of this bottle.

3.1.2.2 makespan

```
int makespan
```

Makespan of this bottle.

3.1.2.3 order

```
int order[MAXJOB]
```

Job order of this bottle.

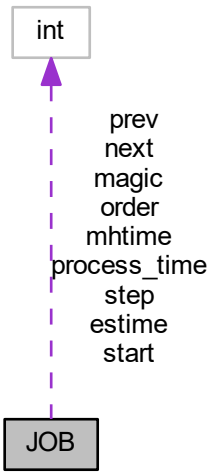
The documentation for this struct was generated from the following file:

- bottle.c

3.2 JOB Struct Reference

```
#include <bottle.h>
```

Collaboration diagram for JOB:



Data Fields

- int estime [MAXMACHINE]
- int mhtime [MAXMACHINE]
- int magic [MAXMACHINE]
- int order [MAXMACHINE]
- int process_time [MAXMACHINE]
- int step [MAXMACHINE]
- int next [MAXMACHINE]
- int prev [MAXMACHINE]
- int start [MAXMACHINE]

3.2.1 Detailed Description

Data representation for a job.

3.2.2 Field Documentation

3.2.2.1 estime

```
int estime[MAXMACHINE]
```

Earlist starting time of this job on each machine. Which is simply the sum of this job's processing times on the machine before [order[machine]] in this jobs prescribed ordering.

3.2.2.2 magic

```
int magic[MAXMACHINE]
```

The number generated and managed by the God in the computer. Every one who changed the name of this feild will be seen as an evil and will be cursed by the God.

3.2.2.3 mhtime

```
int mhtime[MAXMACHINE]
```

Minimum halting time of this job after [machine num]. Which is simply the sum of this job's processing times on the machine after [order[machine]] in this jobs prescribed ordering.

3.2.2.4 next

```
int next[MAXMACHINE]
```

Next job on machine [i].

3.2.2.5 order

```
int order[MAXMACHINE]
```

Required machine order for the job.

3.2.2.6 prev

```
int prev[MAXMACHINE]
```

Previous job blah blah.

3.2.2.7 process_time

```
int process_time[MAXMACHINE]
```

Process time of each machine.

3.2.2.8 start

```
int start[MAXMACHINE]
```

Start time of this job on each machine.

3.2.2.9 step

```
int step[MAXMACHINE]
```

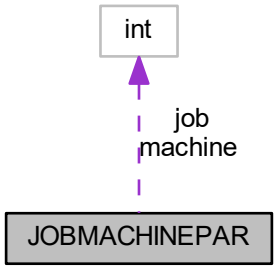
Solution step indexed by machine.

The documentation for this struct was generated from the following file:

- bottle.h

3.3 JOBMACHINEPAR Struct Reference

Collaboration diagram for JOBMACHINEPAR:



Data Fields

- int job
- int machine

3.3.1 Detailed Description

Auxiliary struct to calcuate the makespan.

3.3.2 Field Documentation

3.3.2.1 job

```
int job
```

3.3.2.2 machine

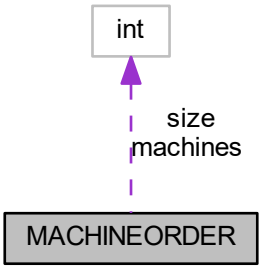
```
int machine
```

The documentation for this struct was generated from the following file:

- eval.c

3.4 MACHINEORDER Struct Reference

Collaboration diagram for MACHINEORDER:



Data Fields

- int size
- int machines [MAXMACHINE]

3.4.1 Detailed Description

Machine order type.

3.4.2 Field Documentation

3.4.2.1 machines

```
int machines[MAXMACHINE]
```

Sequenced machine list.

3.4.2.2 size

```
int size
```

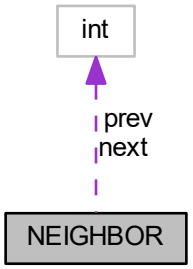
Sequenced machine number.

The documentation for this struct was generated from the following file:

- bottle.c

3.5 NEIGHBOR Struct Reference

Collaboration diagram for NEIGHBOR:



Data Fields

- int next [MAXMACHINE]
- int prev [MAXMACHINE]

3.5.1 Detailed Description

A temporary struct to store a sequence.

3.5.2 Field Documentation

3.5.2.1 next

```
int next[MAXMACHINE]
```

You will be either silly or able to understand the name's mean.

3.5.2.2 prev

```
int prev[MAXMACHINE]
```

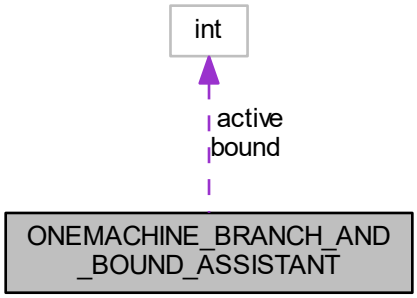
Same as the previous one.

The documentation for this struct was generated from the following file:

- bottle.c

3.6 ONEMACHINE_BRANCH_AND_BOUND_ASSISTANT Struct Reference

Collaboration diagram for ONEMACHINE_BRANCH_AND_BOUND_ASSISTANT:



Data Fields

- int active
- int bound

3.6.1 Detailed Description

Info of node of a branch and bound tree.

3.6.2 Field Documentation

3.6.2.1 active

```
int active
```

Wether this node is active

3.6.2.2 bound

```
int bound
```

See "JOBTYPE" for more info.

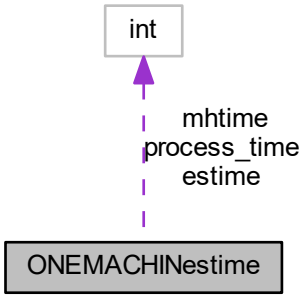
The documentation for this struct was generated from the following file:

- onemachine.c

3.7 ONEMACHINestime Struct Reference

```
#include <bottle.h>
```

Collaboration diagram for ONEMACHINestime:



Data Fields

- int estime [MAXJOB]
- int mhtime [MAXJOB]
- int process_time [MAXJOB]

3.7.1 Detailed Description

Store the time info for every job runs on the same machine.

3.7.2 Field Documentation

3.7.2.1 estime

```
int estime[MAXJOB]
```

See "JOBTYPE" for more info.

3.7.2.2 mhtime

```
int mhtime[MAXJOB]
```

See "JOBTYPE" for more info.

3.7.2.3 process_time

```
int process_time[MAXJOB]
```

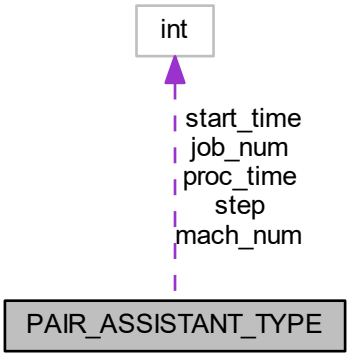
See "JOBTYPE" for more info.

The documentation for this struct was generated from the following file:

- bottle.h

3.8 PAIR_ASSISTANT_TYPE Struct Reference

Collaboration diagram for PAIR_ASSISTANT_TYPE:



Data Fields

- int start_time
- int job_num
- int mach_num
- int proc_time
- int step

3.8.1 Detailed Description

A temporary struct for converting the internal representation of the solution to the format required by those sore-heads.

3.8.2 Field Documentation

3.8.2.1 job_num

```
int job_num
```

Serial number of the job of this node.

3.8.2.2 mach_num

```
int mach_num
```

Serial number of the machine of this node.

3.8.2.3 proc_time

```
int proc_time
```

Process time (a.k.a duration) of this node.

3.8.2.4 start_time

```
int start_time
```

Start time of this node.

3.8.2.5 step

```
int step
```

Serial number of the order of this node in the job.

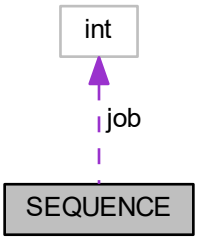
The documentation for this struct was generated from the following file:

- io.c

3.9 SEQUENCE Struct Reference

```
#include <bottle.h>
```

Collaboration diagram for SEQUENCE:



Data Fields

- int job [MAXJOB]

3.9.1 Detailed Description

Job sequences on a machine.

3.9.2 Field Documentation

3.9.2.1 job

```
int job[MAXJOB]
```

Job sequences on a machine.

The documentation for this struct was generated from the following file:

- bottle.h

Chapter 4

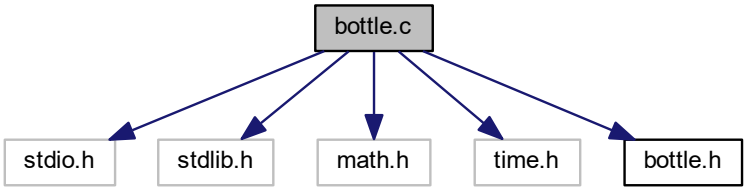
File Documentation

4.1 bottle.c File Reference

Core algorithms.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include "bottle.h"
```

Include dependency graph for bottle.c:



Data Structures

- struct BLIST
- struct MACHINEORDER
- struct NEIGHBOR

Macros

- #define TRY_COUNT 10

Typedefs

- typedef struct BLIST blist_t
- typedef struct MACHINEORDER mo_t
- typedef struct NEIGHBOR neighbor_t

Functions

- static void shifting_bottle_neck (sequence_t *seq, mo_t *machine_order, int *try_time_set)

  The major implementation of the Shifting Bottleneck Procedure, with a backtracing method.
- static void clear_and_backup_seq (sequence_t *seq, int mach, int *save)
- static void clear_seq (sequence_t *seq, int machine)
- static void cp_mo (mo_t *new, mo_t *origin)
- static void cp_seq (sequence_t *new, sequence_t *origin)
- static void cp_neighbor (neighbor_t *new)
- static void re_optimization_phase_1 (sequence_t *seq, mo_t *mo, int *makespan)
- static void set_seq (sequence_t *seq, int mach, int *order)
- static void save_times (void)
- static void re_optimization_phase_2 (sequence_t *seq, mo_t *mo, int *makespan)
- static void restore_neighbor (neighbor_t *old)
- static int critical (int machine, int makespan)
- int eval (sequence_t *seq)

Variables

- int best_makespan = INFINITAS

  Store the best makespan value.

4.1.1 Detailed Description

Core algorithms.

Core algorithms to solve the JSSP.

Author

Name1e5s

4.1.2 Macro Definition Documentation

4.1.2.1 TRY_COUNT

```
#define TRY_COUNT 10
```

4.1.3 Typedef Documentation

4.1.3.1 blist_t

```
typedef struct BLIST blist_t
```

Store the bottle informnation.

4.1.3.2 mo_t

```
typedef struct MACHINEORDER mo_t
```

Machine order type.

4.1.3.3 neighbor_t

```
typedef struct NEIGHBOR neighbor_t
```

A temporary struct to store a sequence.

4.1.4 Function Documentation

4.1.4.1 clear_and_backup_seq()

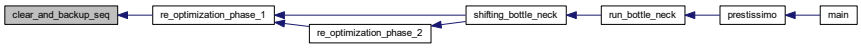
```
static void clear_and_backup_seq (
            sequence_t * seq,
            int machine,
            int * save )  [inline], [static]
```

Store current sequence of machine N in the given address. Then just clear the sequence.

Parameters

seq	Sequence to be cleared.
machine	Current machine number.
save	Address to save the old sequence. A NULL address means the old sequence won't be stored.

Here is the caller graph for this function:



4.1.4.2 clear_seq()

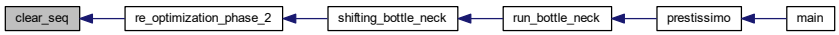
```
static void clear_seq (
            sequence_t * seq,
            int machine )  [inline], [static]
```

Clear the sequence.

Parameters

seq	Sequence to be cleared.
machine	Current machine number.

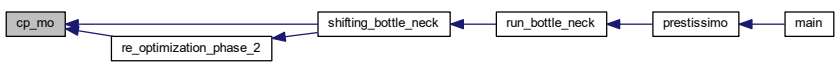
Here is the caller graph for this function:



4.1.4.3 cp_mo()

```
static void cp_mo (
            mo_t * new,
            mo_t * origin )  [inline], [static]
```

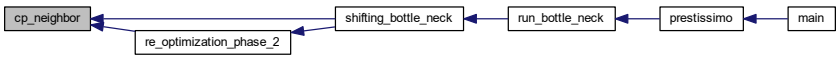
Copy the origin machine order to new. Here is the caller graph for this function:



4.1.4.4 cp_neighbor()

```
static void cp_neighbor (
            neighbor_t * new )  [inline], [static]
```

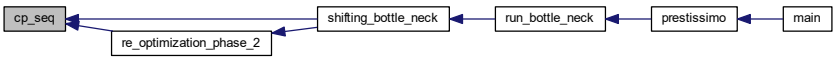
Store neighbor to a neighbor_t varible. Here is the caller graph for this function:



4.1.4.5 cp_seq()

```
static void cp_seq (
            sequence_t * new,
            sequence_t * origin )  [inline], [static]
```

Copy the origin sequence order to new. Here is the caller graph for this function:



4.1.4.6 critical()

```
static int critical (
            int machine,
            int makespan )  [inline], [static]
```

Test wether the machine is the critical machine, which means the end of the procedure of this machine is also the end of all the operations.

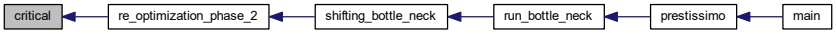
Parameters

machine	Machine number to be tested.
makespan	The given makespan

Returns

If the machine is the critical machine, return 1. Else return 0.

Here is the caller graph for this function:



4.1.4.7 eval()

```
int eval (
            sequence_t * seq )
```

Evaluate the makespan of the given sequence.

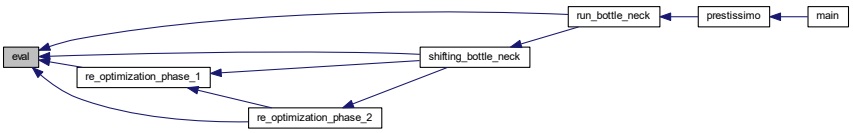
Parameters

seq	The sequence of job.

Returns

The makespan of the sequence.

Here is the caller graph for this function:



4.1.4.8 re_optimization_phase_1()

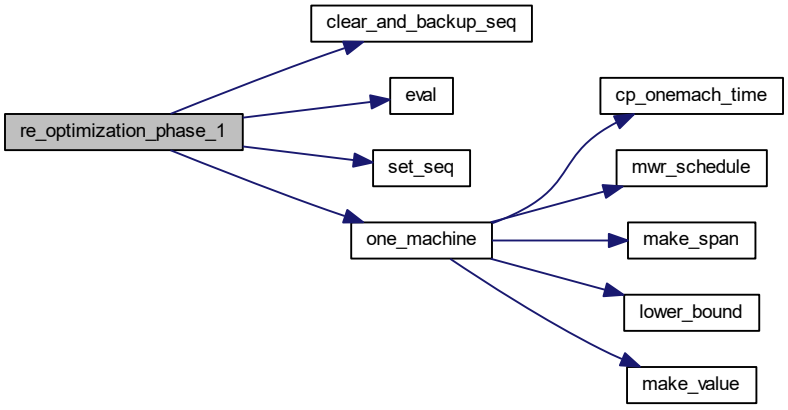
```
static void re_optimization_phase_1 (
            sequence_t * seq,
            mo_t * machine_order,
            int * makespan )  [inline], [static]
```

The re-optimization... Phase 1

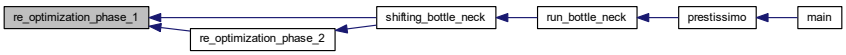
Parameters

seq	The sequence
machine_order	Machine order
makespan	Current makespan

Here is the call graph for this function:



Here is the caller graph for this function:



4.1.4.9 re_optimization_phase_2()

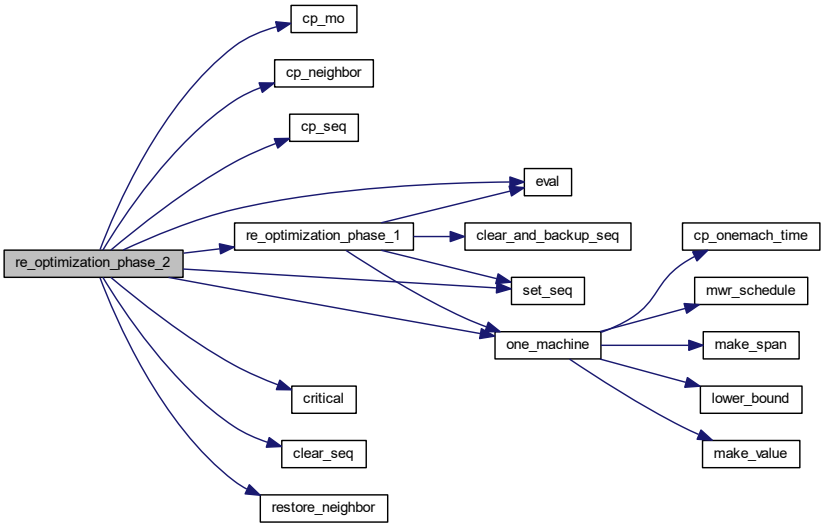
```
static void re_optimization_phase_2 (
            sequence_t * seq,
            mo_t * machine_order,
            int * makespan )  [inline], [static]
```

The re-optimization... Phase 2

Parameters

seq	The sequence
machine_order	Machine order
makespan	Current makespan

Here is the call graph for this function:

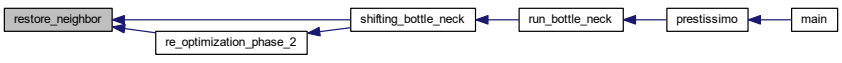


Here is the caller graph for this function:

4.1.4.10 restore_neighbor()

```
static void restore_neighbor (
            neighbor_t * old )  [inline], [static]
```

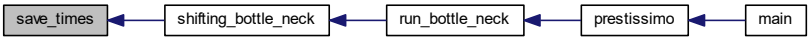
Load neighbor from a neighbor_t varible. Here is the caller graph for this function:



4.1.4.11 save_times()

```
static void save_times (
            void  )  [inline], [static]
```

Save current start time of each operation. Here is the caller graph for this function:



4.1.4.12 set_seq()

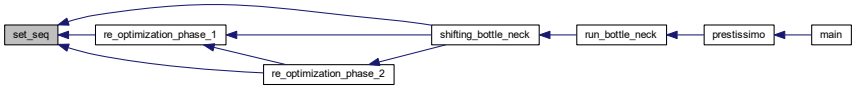
```
static void set_seq (
            sequence_t * seq,
            int machine,
            int * order )  [inline], [static]
```

Set sequence by the given order.

Parameters

seq	Sequence to be set.
machine	The machine which the sequence relies on.
order	The given order.

Here is the caller graph for this function:



4.1.4.13 shifting_bottle_neck()

```
static void shifting_bottle_neck (
            sequence_t * seq,
            mo_t * machine_order,
            int * try_time_set )  [inline], [static]
```

The major implementation of the Shifting Bottleneck Procedure, with a backtracing method.

The basic idea of the algorithm can be described as follows: It sequences the machines one by one successively, taking each time the machine identified as a bottleneck among the machine not yet sequenced. Every time after a new machine is sequenced, all previously sequenced sequence will be locally re-optimized. Bottleneck identification and the local re-optimization are both based on solving a one machine scheduling problem, which is more easy than the JSSP. In this implementation a backtracing trick is introduced to improve the quality of the solution, which give us a method to use a slightly more time to run the basic shifting bottleneck procedure more times.

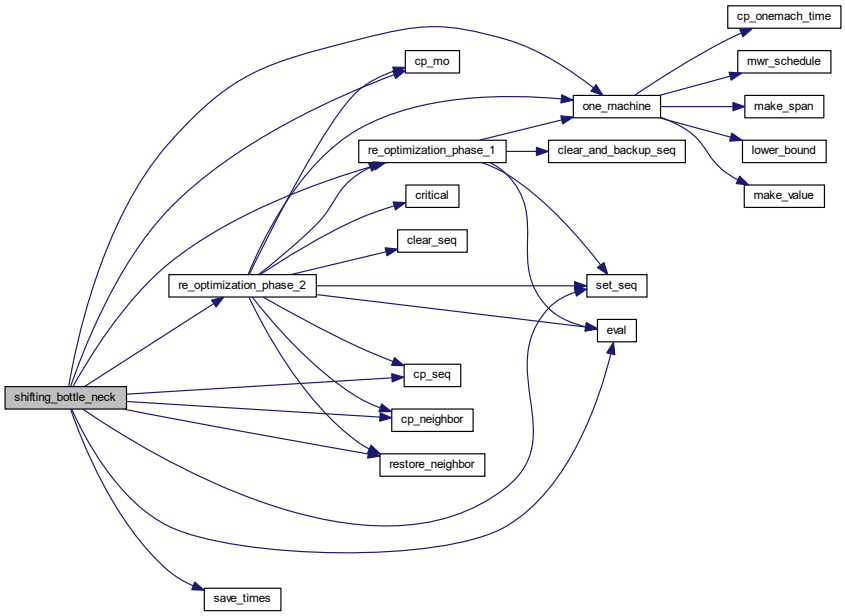
Parameters

seq	The given sequence list. Will be updated when find a better makespan.
machine_order	Machine order.
try_time_set	Backtracing depth set.

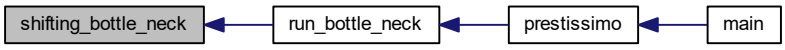
Returns

When the procedure is done. You should find the start time of the solution at the "start" field of the struct array job.

Here is the call graph for this function:



Here is the caller graph for this function:



4.1.5 Variable Documentation

4.1.5.1 best_makespan

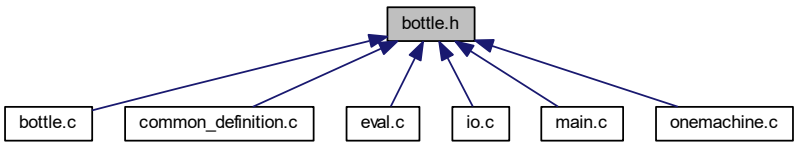
```
best_makespan = INFINITAS
```

Store the best makespan value.

4.2 bottle.h File Reference

Header file for the whole project.

This graph shows which files directly or indirectly include this file:



Data Structures

- struct JOB
- struct SEQUENCE
- struct ONEMACHINestime

Macros

- #define MAXJOB 30
- #define MAXMACHINE 30
- #define INFINITAS 0x7fffffff
- #define MAX(a, b) ((a) > (b) ? (a) : (b))

Typedefs

- typedef struct JOB job_t
- typedef struct SEQUENCE sequence_t
- typedef struct ONEMACHINestime onemach_times_t

Functions

- void prestissimo (void)
- void run_bottle_neck (void)
- int one_machine (onemach_times_t one, int *bestorder)

Variables

- job_t job [MAXJOB]
- int job_size
- int machine_size
- int terminate_flag

4.2.1 Detailed Description

Header file for the whole project.

A Simple Old-fashion Implementation Of The Well-known Shifting Bottleneck Procedure For Job Shop Scheduling Problem(JSSP).The codes are based on "The Shifting Bottleneck Procedure for Job Shop Scheduling" by J. Adams et al.

Author

Name1e5s

4.2.2 Macro Definition Documentation

4.2.2.1 INFINITAS

```
#define INFINITAS 0x7fffffff
```

A integer that can be seen as infinity – should be bigger than the biggest makespan of all the instances. Hence, 0x7fffffff (a.k.a INT_MAX) is a good choice

4.2.2.2 MAX

```
#define MAX(
             a,
             b ) ((a) > (b) ?  (a) :  (b))
```

A regular macro that returns the bigger value bewteen a and b.

4.2.2.3 MAXJOB

```
#define MAXJOB 30
```

The most jobs this program can handle.

4.2.2.4 MAXMACHINE

```
#define MAXMACHINE 30
```

The most machines this program can handle.

4.2.3 Typedef Documentation

4.2.3.1 job_t

```
typedef struct JOB job_t
```

Data representation for a job.

4.2.3.2 onemach_times_t

```
typedef struct ONEMACHINestime onemach_times_t
```

Store the time info for every job runs on the same machine.

4.2.3.3 sequence_t

```
typedef struct SEQUENCE sequence_t
```

Job sequences on a machine.

4.2.4 Function Documentation

4.2.4.1 one_machine()

```
int one_machine (
            onemach_times_t one,
            int * bestorder )
```

The one-machine sequencing algorithm from "The one-machine sequencing problem" by Jacques Carlier.

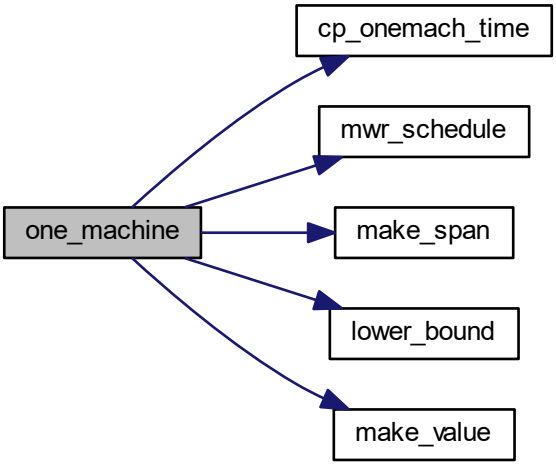
Parameters

one	Representation of the machine.
bestorder	Best job order

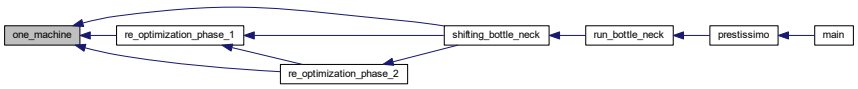
Returns

makespan

Here is the call graph for this function:



Here is the caller graph for this function:



4.2.4.2 prestissimo()

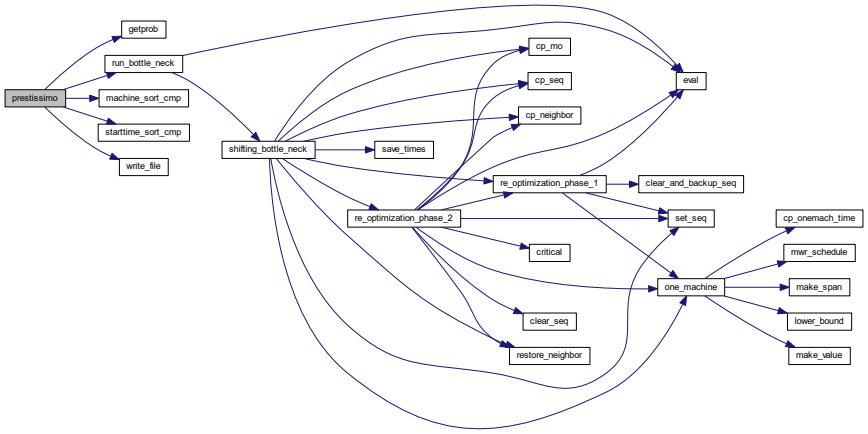
```
void prestissimo (
            void  )
```

Convert internal solution representation structure to the format required by those nitpickers and print it.

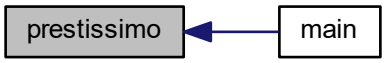
Parameters

filename	Instance file path

Here is the call graph for this function:



Here is the caller graph for this function:

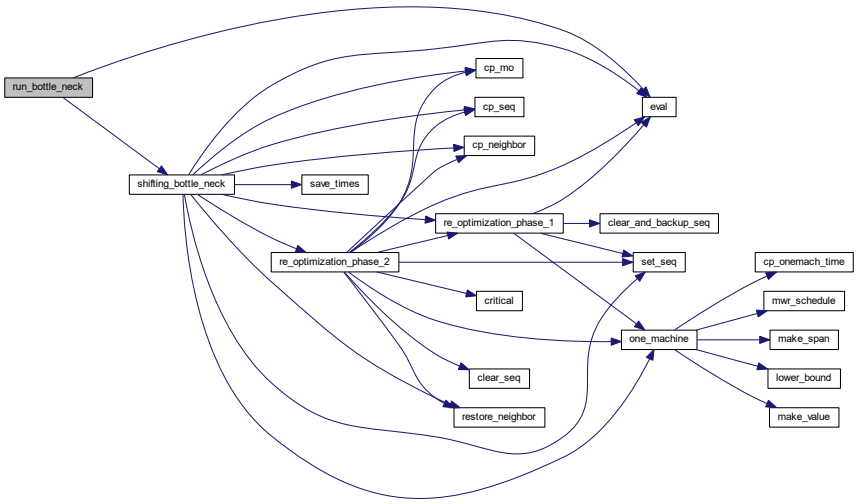


4.2.4.3 run_bottle_neck()

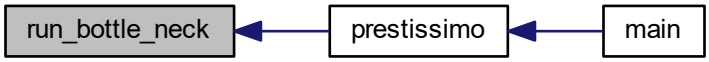
```
void run_bottle_neck (
            void  )
```

Driver of the Shifting Bottleneck Procedure We can change here to have a balance bewteen run time and

makespan...Here is the call graph for this function:



Here is the caller graph for this function:



4.2.5 Variable Documentation

4.2.5.1 job

`job`

Data representation of all the jobs. All operations runs on this varible.

4.2.5.2 job_size

`job_size`

Job number in this instance.

4.2.5.3 machine_size

`machine_size`

Machine number in this instance.

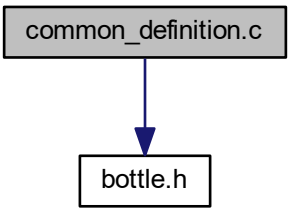
4.2.5.4 terminate_flag

`terminate_flag`

Should we stop???

4.3 common_definition.c File Reference

`#include "bottle.h"`
Include dependency graph for common_definition.c:



Variables

- job_t job [MAXJOB]
- int job_size
- int machine_size
- int terminate_flag

4.3.1 Variable Documentation

4.3.1.1 job

`job_t job[MAXJOB]`

Data representation of all the jobs. All operations runs on this varible.

4.3.1.2 job_size

```
int job_size
```

Job number in this instance.

4.3.1.3 machine_size

```
int machine_size
```

Machine number in this instance.

4.3.1.4 terminate_flag

```
int terminate_flag
```

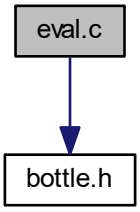
Should we stop???

4.4 eval.c File Reference

makespan

```
#include "bottle.h"
```

Include dependency graph for eval.c:



Data Structures

- struct JOBMACHINEPAR

Typedefs

- typedef struct JOBMACHINEPAR job_machine_t

Functions

- int eval (sequence_t *seq)

Variables

- int magicnum = 0

4.4.1 Detailed Description

makespan

Function to calcuate makespan.

Author

Name1e5s

4.4.2 Typedef Documentation

4.4.2.1 job_machine_t

```
typedef struct JOBMACHINEPAR job_machine_t
```

Auxiliary struct to calcuate the makespan.

4.4.3 Function Documentation

4.4.3.1 eval()

```
int eval (
            sequence_t * seq )
```

Evaluate the makespan of the given sequence.

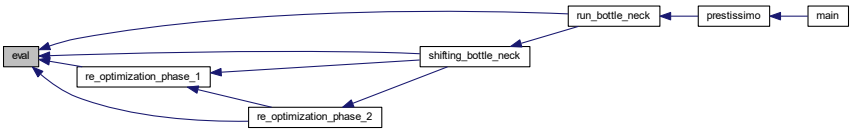
Parameters

seq	The sequence of job.

Returns

The makespan of the sequence.

Here is the caller graph for this function:



4.4.4 Variable Documentation

4.4.4.1 magicnum

```
magicnum = 0
```

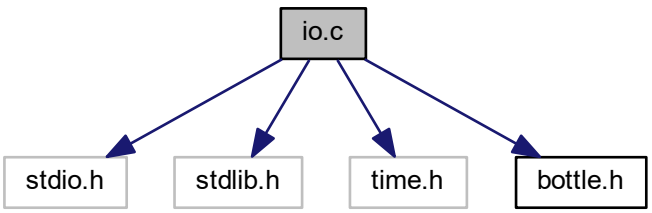
The number generated and managed by the God in the computer. Every one who changed the name of this feild will be seen as an evil and will be cursed by the God.

4.5 io.c File Reference

IO.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "bottle.h"
```

Include dependency graph for io.c:



Data Structures

- struct PAIR_ASSISTANT_TYPE

Typedefs

- typedef struct PAIR_ASSISTANT_TYPE pair_ass_t

Functions

- static void getprob (void)
- static void write_file (char *file_name, int ans, pair_ass_t *pairs, float times)
- int machine_sort_cmp (const void *a, const void *b)
- int starttime_sort_cmp (const void *a, const void *b)
- void prestissimo (void)

Variables

- int best_makespan

  Store the best makespan value.

4.5.1 Detailed Description

IO.

Functions to handle input and output.

Author

Name1e5s

4.5.2 Typedef Documentation

4.5.2.1 pair_ass_t

```
typedef struct PAIR_ASSISTANT_TYPE pair_ass_t
```

A temporary struct for converting the internal representation of the solution to the format required by those sore-heads.

4.5.3 Function Documentation

4.5.3.1 getprob()

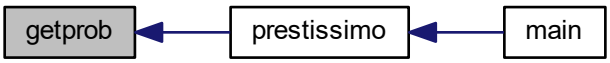
```
static void getprob (
            void  )  [inline], [static]
```

Read instance file.

Parameters

file_name	Instance file path

Here is the caller graph for this function:



4.5.3.2 machine_sort_cmp()

```
int machine_sort_cmp (
            const void * a,
            const void * b )
```

Function to compare machine number of two pairs for qsort.

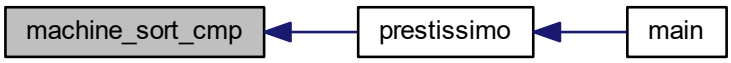
Parameters

a	The first pair.
b	The second pair.

Returns

If machine number of a is lesser than b, then return a positive value, else return a non-positive value.

Here is the caller graph for this function:



4.5.3.3 prestissimo()

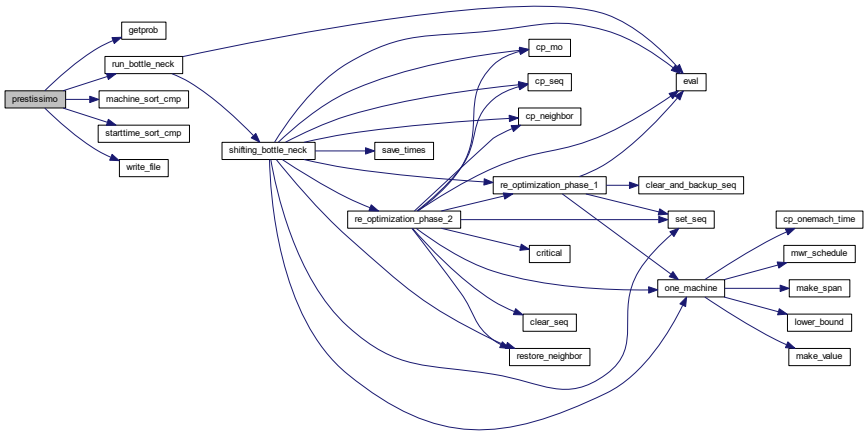
```
void prestissimo (
            void  )
```

Convert internal solution representation structure to the format required by those nitpickers and print it.

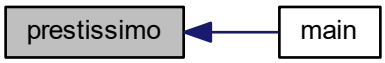
Parameters

filename	Instance file path

Here is the call graph for this function:



Here is the caller graph for this function:



4.5.3.4 starttime_sort_cmp()

```
int starttime_sort_cmp (
            const void * a,
            const void * b )
```

Function to compare start time of two pairs for qsort.

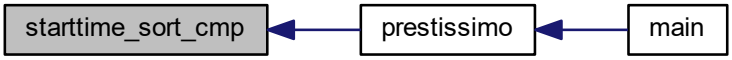
Parameters

a	The first pair.
b	The second pair.

Returns

If start time of a is lesser than b, then return a positive value, else return a non-positive value.

Here is the caller graph for this function:



4.5.3.5 write_file()

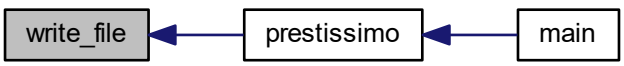
```
static void write_file (
            char * file_name,
            int ans,
            pair_ass_t * pairs,
            float times )  [inline], [static]
```

Print result to file...

Parameters

file_name	Instance file path
pairs	Pair to be printed...

Here is the caller graph for this function:



4.5.4 Variable Documentation

4.5.4.1 best_makespan

```
int best_makespan
```

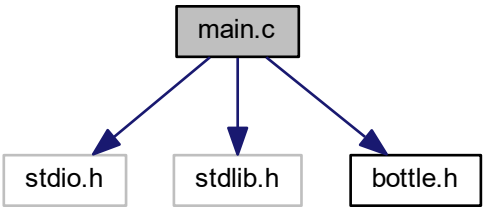
Store the best makespan value.

4.6 main.c File Reference

Enterpoint.

```
#include <stdio.h>
#include <stdlib.h>
#include "bottle.h"
```

Include dependency graph for main.c:



Functions

- int main (int argc, char **argv)

4.6.1 Detailed Description

Enterpoint.

Enterpoint of the program.

Author

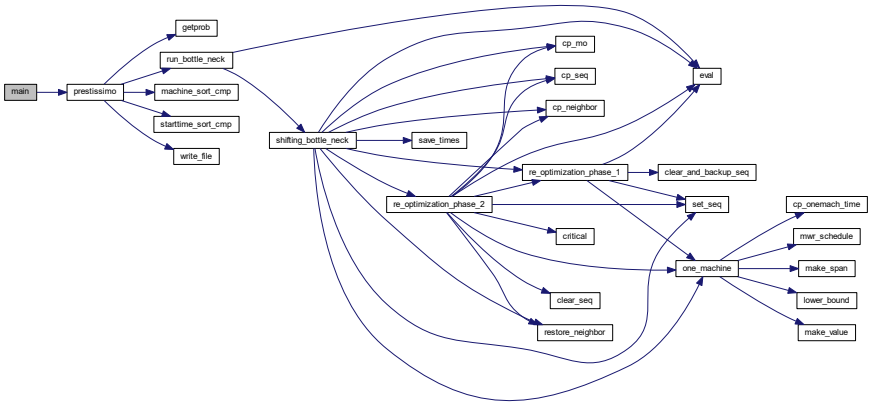
Name1e5s

4.6.2 Function Documentation

4.6.2.1 main()

```
int main (
            int argc,
            char ** argv )
```

Here is the call graph for this function:

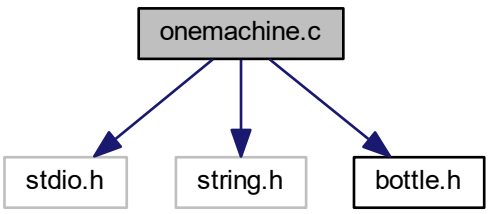


4.7 onemachine.c File Reference

One-machine sequencing.

```
#include <stdio.h>
#include <string.h>
#include "bottle.h"
```

Include dependency graph for onemachine.c:



Data Structures

- struct ONEMACHINE_BRANCH_AND_BOUND_ASSISTANT

Macros

- #define ONEMACH_BBNODES 300

Typedefs

- typedef struct ONEMACHINE_BRANCH_AND_BOUND_ASSISTANT onemach_bb_ass_t

Functions

- static void cp_onemach_time (onemach_times_t *new, onemach_times_t *origin)
- static void mwr_schedule (onemach_times_t one, int *order)
- static int lower_bound (onemach_times_t one, int *jset, int jset_size)
- static int make_span (onemach_times_t one, int *order, int *jset, int *jset_size, int *cjob, int *pjob, int *make)
- static int make_value (onemach_times_t one, int *order)
- int one_machine (onemach_times_t one, int *bestorder)

4.7.1 Detailed Description

One-machine sequencing.

One-machine Sequencing Algorithm from Jacques Carlier

Author

TJenica

4.7.2 Macro Definition Documentation

4.7.2.1 ONEMACH_BBNODES

```
#define ONEMACH_BBNODES 300
```

Nodes number of the branch and bound tree to solve the one machine sequencing problem.

4.7.3 Typedef Documentation

4.7.3.1 onemach_bb_ass_t

```
typedef struct ONEMACHINE_BRANCH_AND_BOUND_ASSISTANT onemach_bb_ass_t
```

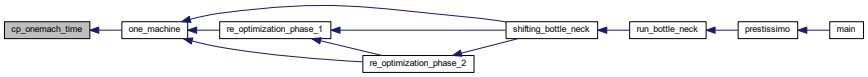
Info of node of a branch and bound tree.

4.7.4 Function Documentation

4.7.4.1 cp_onemach_time()

```
static void cp_onemach_time (
            onemach_times_t * new,
            onemach_times_t * origin )  [inline], [static]
```

Copy origin onemach_times struct to new By practice, change from memcpy to just write what we want to do is very important... Here is the caller graph for this function:



4.7.4.2 lower_bound()

```
static int lower_bound (
            onemach_times_t one,
            int * job_set,
            int job_set_size )  [inline], [static]
```

Find the lower bound of the given machine on the given job order.

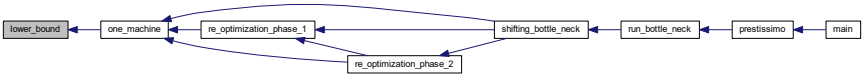
Parameters

one	The representation of the given machine.
job_set	The set of job.
job_set_size	The size of job_set

Returns

Lowerbound of the machine. Which is just the sum of minimum estime and minimum mhtime and the sum of all the process time.

Here is the caller graph for this function:



4.7.4.3 make_span()

```
static int make_span (
             onemach_times_t one,
             int * order,
             int * job_set,
             int * job_set_size,
             int * critical_job_order,
             int * terminate_job_order,
             int * make )  [inline], [static]
```

Test if the job order is feasible and compute the make_span.

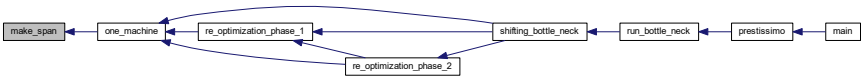
Parameters

one	The representation of the given machine.
order	The given job order.
job_set	The set of job on the machine.
job_set_size	The size of job_set.
critical_job_order	
terminate_job_order	
make	The make_span.

Returns

If the order is OK return 1,else return 0.

Here is the caller graph for this function:



4.7.4.4 make_value()

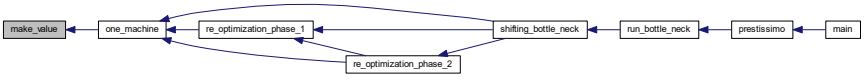
```
static int make_value (
             onemach_times_t one,
             int * order )  [inline], [static]
```

Compute the makespan of the given job order.

Parameters

one	Representation of the machine. order The given job order.

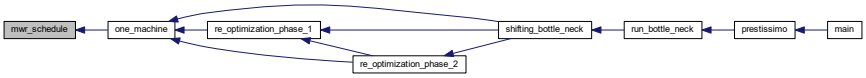
Here is the caller graph for this function:



4.7.4.5 mwr_schedule()

```
static void mwr_schedule (
             onemach_times_t one,
             int * order )  [inline], [static]
```

Algorithm to find the most work remaining schedule by Schrage Here is the caller graph for this function:



4.7.4.6 one_machine()

```
int one_machine (
             onemach_times_t one,
             int * bestorder )
```

The one-machine sequencing algorithm from "The one-machine sequencing problem" by Jacques Carlier.

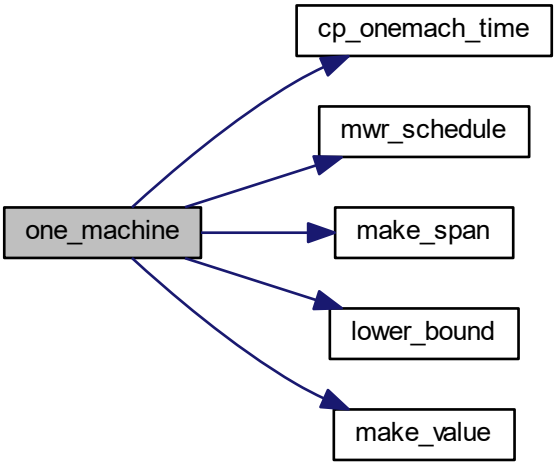
Parameters

one	Representation of the machine.
bestorder	Best job order

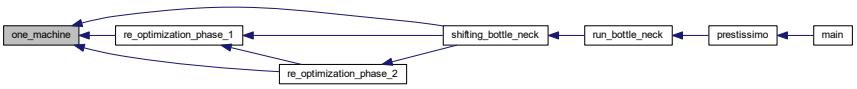
Returns

makespan

Here is the call graph for this function:



Here is the caller graph for this function:



Index

active
ONEMACHINE_BRANCH_AND_BOUND_ASSI↩
STANT, 12

BLIST, 5
machine, 5
makespan, 6
order, 6
best_makespan
bottle.c, 27
io.c, 41
blist_t
bottle.c, 19
bottle.c, 17
best_makespan, 27
blist_t, 19
clear_and_backup_seq, 19
clear_seq, 20
cp_mo, 20
cp_neighbor, 20
cp_seq, 21
critical, 21
eval, 22
mo_t, 19
neighbor_t, 19
re_optimization_phase_1, 22
re_optimization_phase_2, 23
restore_neighbor, 24
save_times, 24
set_seq, 25
shifting_bottle_neck, 25
TRY_COUNT, 18
bottle.h, 27
INFINITAS, 28
job, 32
job_size, 32
job_t, 29
MAXJOB, 28
MAXMACHINE, 28
MAX, 28
machine_size, 32
one_machine, 29
onemach_times_t, 29
prestissimo, 30
run_bottle_neck, 31
sequence_t, 29
terminate_flag, 33
bound
ONEMACHINE_BRANCH_AND_BOUND_ASSI↩
STANT, 12

clear_and_backup_seq
bottle.c, 19
clear_seq
bottle.c, 20
common_definition.c, 33
job, 33
job_size, 33
machine_size, 34
terminate_flag, 34
cp_mo
bottle.c, 20
cp_neighbor
bottle.c, 20
cp_onemach_time
onemachine.c, 44
cp_seq
bottle.c, 21
critical
bottle.c, 21

estime
JOB, 7
ONEMACHINestime, 13
eval
bottle.c, 22
eval.c, 35
eval.c, 34
eval, 35
job_machine_t, 35
magicnum, 36

getprob
io.c, 37

INFINITAS
bottle.h, 28
io.c, 36
best_makespan, 41
getprob, 37
machine_sort_cmp, 38
pair_ass_t, 37
prestissimo, 38
starttime_sort_cmp, 39
write_file, 40

JOBMACHINEPAR, 8
job, 9
machine, 9
JOB, 6
estime, 7

magic, 7
mhtime, 7
next, 7
order, 7
prev, 8
process_time, 8
start, 8
step, 8
job
bottle.h, 32
common_definition.c, 33
JOBMACHINEPAR, 9
SEQUENCE, 16
job_machine_t
eval.c, 35
job_num
PAIR_ASSISTANT_TYPE, 14
job_size
bottle.h, 32
common_definition.c, 33
job_t
bottle.h, 29

lower_bound
onemachine.c, 44

MACHINEORDER, 9
machines, 10
size, 10
MAXJOB
bottle.h, 28
MAXMACHINE
bottle.h, 28
MAX
bottle.h, 28
mach_num
PAIR_ASSISTANT_TYPE, 14
machine
BLIST, 5
JOBMACHINEPAR, 9
machine_size
bottle.h, 32
common_definition.c, 34
machine_sort_cmp
io.c, 38
machines
MACHINEORDER, 10
magic
JOB, 7
magicnum
eval.c, 36
main
main.c, 42
main.c, 41
main, 42
make_span
onemachine.c, 45
make_value
onemachine.c, 45
makespan
BLIST, 6
mhtime
JOB, 7
ONEMACHINestime, 13
mo_t
bottle.c, 19
mwr_schedule
onemachine.c, 46

NEIGHBOR, 10
next, 11
prev, 11
neighbor_t
bottle.c, 19
next
JOB, 7
NEIGHBOR, 11

ONEMACH_BBNODES
onemachine.c, 43
ONEMACHINE_BRANCH_AND_BOUND_ASSISTA↩
NT, 11
active, 12
bound, 12
ONEMACHINestime, 12
estime, 13
mhtime, 13
process_time, 13
one_machine
bottle.h, 29
onemachine.c, 46
onemach_bb_ass_t
onemachine.c, 43
onemach_times_t
bottle.h, 29
onemachine.c, 42
cp_onemach_time, 44
lower_bound, 44
make_span, 45
make_value, 45
mwr_schedule, 46
ONEMACH_BBNODES, 43
one_machine, 46
onemach_bb_ass_t, 43
order
BLIST, 6
JOB, 7

PAIR_ASSISTANT_TYPE, 14
job_num, 14
mach_num, 14
proc_time, 15
start_time, 15
step, 15
pair_ass_t
io.c, 37
prestissimo
bottle.h, 30

io.c, 38
prev
JOB, 8
NEIGHBOR, 11
proc_time
PAIR_ASSISTANT_TYPE, 15
process_time
JOB, 8
ONEMACHINestime, 13

re_optimization_phase_1
bottle.c, 22
re_optimization_phase_2
bottle.c, 23
restore_neighbor
bottle.c, 24
run_bottle_neck
bottle.h, 31

SEQUENCE, 15
job, 16
save_times
bottle.c, 24
sequence_t
bottle.h, 29
set_seq
bottle.c, 25
shifting_bottle_neck
bottle.c, 25
size
MACHINEORDER, 10
start
JOB, 8
start_time
PAIR_ASSISTANT_TYPE, 15
starttime_sort_cmp
io.c, 39
step
JOB, 8
PAIR_ASSISTANT_TYPE, 15

TRY_COUNT
bottle.c, 18
terminate_flag
bottle.h, 33
common_definition.c, 34

write_file
io.c, 40