Concorde README and Installation guide

This README file is part of the 03.12.19 release of Concorde. More information about Concorde is available from the Concorde page.

Concorde is a computer code for the traveling salesman problem (TSP) and some related network optimization problems. The code is written in the ANSI C programming language and it is available for academic research use; for other uses, contact bico@uwaterloo.ca for licensing options.

Concorde's TSP solver has been used to obtain the optimal solutions to all TSPLIB instances having up to 15,112 cities.

The Concorde callable library includes over 700 functions permitting users to create specialized codes for TSP-like problems. All Concorde functions are thread-safe for programming in shared-memory parallel environments; the main TSP solver includes code for running over networks of unix workstations. Documentation for the callable library can be found at the web page Concorde Documentation.

The rest of this file contains instructions for unpacking, configuring, and compiling Concorde.

Concorde is distributed as a gzipped tar file. To unpack, first gunzip the downloaded file, then use tar to unpack the archive:

```
$ gunzip co031219.tgz
$ tar xvf co031219.tar
```

Unpacking the distribution create a directory, "concorde", and several subdirectories. To configure, cd into the concorde directory, and run the configure script:

```
$ cd concorde
$ ./configure
```

configure supports several command-line options:

`--help` Prints a list of all options configure supports. The last options are the most pertinent.

`--with-qsopt=DIR` Indicates to use the QSopt LP solver. DIR should be the full path name (not the relative path name) for a directory containing both qsopt.a and qsopt.h. QSopt is available for no charge at QSopt Home.

`--with-cplex=DIR` Indicates to use the CPLEX LP solver. DIR should be the full path name (not the relative path name) for a directory containing both libcplex.a and and cplex.h. (If these are in different directories then you will need to edit the Makefiles in several of the concorde directories to include the correct search path with -I.) By default this option supports CPLEX 8.0. For earlier versions use with-cplex6, with-cplex5, or with-cplex4. Depending on the version of cplex, you may need to include the -lpthreads option on the linking step in TSP/Makefile (add this to the end of the line defining LIBFLAGS).

`--enable-ccdefaults` Indicates that a set of default compiler and loader flags should be used. We use this flag during development to enable useful optimization and warning options on our development platforms. They may or may not be useful for you. See also the discussion of CC, CFLAGS, CPPFLAGS, and LDFLAGS below.

`--enable-debugger` Overrides the optimization options set by --enable-ccdefaults to support additional debugging.

configure is also affected by several environment variables. CC controls which c compiler to use, CPPFLAGS specifies the C preprocessor flags to use, CFLAGS specifies the C compiler flags to use, and LDFLAGS

specifies loader flags to use. configure generates defaults for all of these variables, by default using "gcc -g -O2" if gcc is available, and "cc -g" otherwise. For better performance of the concorde package, set CFLAGS to "-g -O3" if compiling with gcc, and a strong optimization flag for your compiler otherwise (typically "-O3", "-O2", or "-xO2"). The method for setting a environment variable depends on the type of shell you are using. You can usually find out which kind of shell you are using by typing "echo $SHELL". Otherwise, try one method and if it doesn't work, try the other. If you are using sh, bash, ksh, ash, or some other Bourne-like shell, you can simply set the variables on the command line, for example: $ CC="gcc" CFLAGS="-g -O3" ./configure

If you are using csh, tcsh, or some other C-shell, you set the variables by using setenv, for example: $ setenv CC "gcc" $ setenv CFLAGS "-g -O3" $ ./configure

configure determines many features about your operating system and configuration which it uses to control concorde's compilation. After configure has run, you should look through the file INCLUDE/config.h to make sure it has made reasonable choices. Then, you can compile concorde by using make: $ make

This builds the concorde library (concorde.a), header file (concorde.h), and several executable programs, including:

- `TSP/concorde` the TSP solver
- `LINKERN/linkern` the Lin-Kernighan TSP heuristic
- `EDGEGEN/edgegen` generates edge sets
- `FMATCH/fmatch` solves fractional 2-matching problems

NOTE that to build the concorde TSP solver (for the exact solution of TSPs), you must specify an LP solver in the configure step (either QSopt for CPLEX).

A short help menu for each of the executable codes can be obtained by executing the code without any option, for example "concorde". A typical way to run the concorde solver is to use: "concorde myprob.tsp" where myprob.tsp is a TSP instance in TSPLIB format. To test the concorde solver run "concorde -s 99 -k 100" (this should generate and solve a random geometric TSP on 100 points).

If you are using Gnu make, then you can separate the build directory from the source directory. Simply make the build directory, cd into it, and then run the configure from the source directory. If for some reason configure cannot figure out where the source directory is, specify it with the "--srcdir=DIR" command line argument. A complete example of unpacking, configuring, and compiling in a separate build directory, is:

```
$ gunzip co031219.tgz
$ tar xvf co031219.tar
$ mkdir concorde_build
$ cd concorde_build
$ ../concorde/configure
$ make
```