of the parallel allocation. As the GAMS model itself does not take this aspect into consideration, it has been incorporated in the pre-processing part of the solution.

When assigning exchanges the algorithm does not check for feasibility, but tries to assign difficult exchanges at the inbound end of the streams. In other words, if a hot stream H1 exchanges heat with two cold streams, C1 and C2, the cold stream with the highest outbound temperature receives the hottest part of H1. Likewise, if a cold stream C1 exchanges heat with two hot streams, H1 and H2, it assigns the hot stream with the lowest outbound temperature at the inbound end of C1. This assures that the most demanding exchanges receive the highest quality heat for any given stream.

Hopefully this the method will be able to create more feasible or near-feasible starting solutions. While this way of allocating mass flow is superior to the serial/parallel mCp generator in its use of physical insight when setting flow pattern, it has many imperfections. Exchanger temperatures play an important part when setting mCp flow, but the heat loads and mass flows themselves are of equal importance. When using only temperatures it is easy to choose solutions that may at first sight appear clever, but that may in fact render the starting solution in the non-feasible region.

As stated later in chapter seven, the solvers employed in this study is more likely to return a feasible answer given a feasible or near-feasible starting point. As the lack of possibility to include parallel heat exchange in many cases would restrain this module from obtaining feasible inputs, it became important developing a more advanced mCp module.

5.4 The combinatorial mCp generator

The combinatorial mCp module is the final module created in order to obtain starting values for mCp in this study. It is separated from its two predecessor modules by its complexity and its use of temperatures, heat loads and mass flow when allocating stream pattern. As the name implies it makes use of both serial and parallel configurations. The basic principles governing the module is described in the following flow sheet:

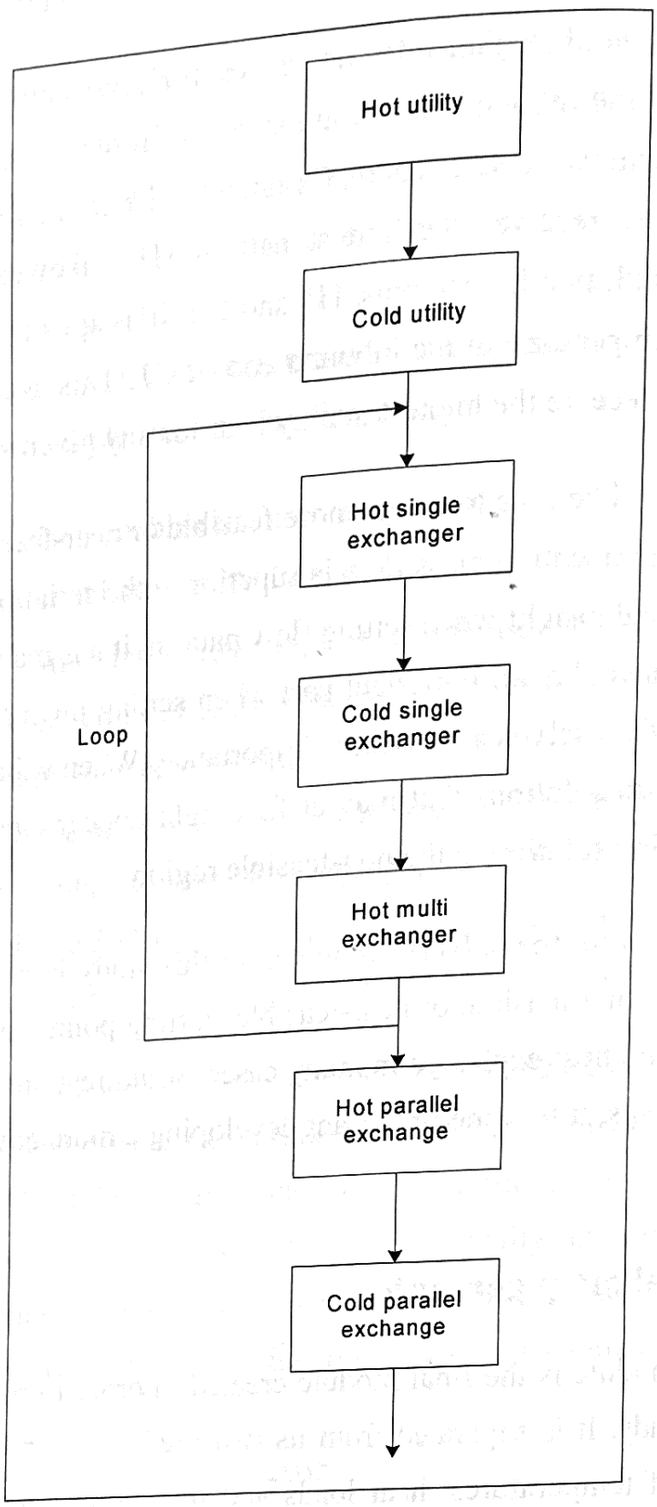

Figure 5.1- The combinatorial mCp generator

The initial step of this module is to allocate utilities. These streams are often either v
cold or very hot streams. It is therefore often natural to place these streams at the o
bound end of the exchanging stream. A cold utility will in many cases have the cold
temperatures in a given HENS problem, and experience show that the best solution
often to place it as the last heat node in its opposite hot stream. Similar measu
should be taken when dealing with hot utilities. Allowing the hot utility to heat the co
stream as the final heat node will in many cases generate a more cost efficient netwo

than otherwise (T.Gundersen, personal communication April 04, 2006). A check is performed in order to see if the placing of the heat node is feasible before it is assigned. Upon assigning, the heat node and stream location is flagged (marked as done) and left out of further processing. If the exchange is infeasible, the algorithm skips the procedure and continues to the next task.

After assigning utility exchangers, the module checks single exchanger streams. There is no physical background for handling these exchangers prior to multiple exchanger streams, but this has been implemented to simplify programming. The module checks all single exchangers for feasible exchanges with all other streams. In order to avoid later feasibility problems due to the assigning of these streams, only exchanges that makes use of the lowest quality heat is accepted. This means that if H1 is a one-exchanger stream that exchanges heat with C1, it will only assign the exchange if it is possible to place the exchange at the hot end of C1. If successful, this will reduce the high quality heat needed when assigning other streams that exchange heat with C1. An example is included in order to further illustrate the concept.

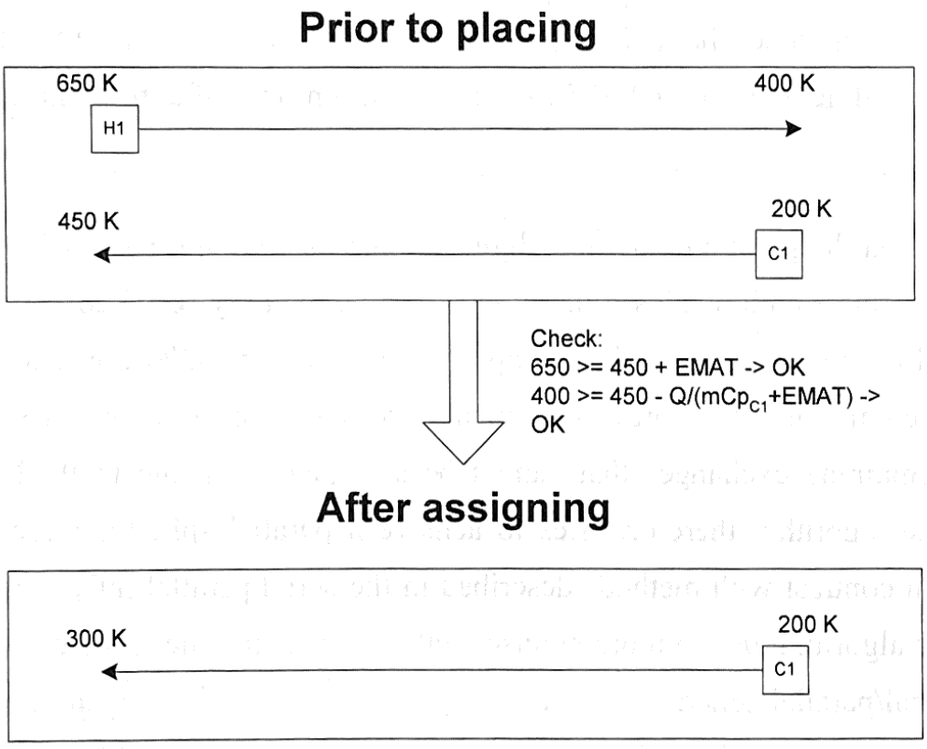

Figure 5.2 - Assigning H1 to C1

Assigning streams in this manor ensures that the possibility of feasible exchanges throughout the problem increases with each assignment. For the example described above, stream C1 has a temperature span that is in a lower region after the assignment to stream H1 than before. This makes C1 easier to handle when facing exchanges with other hot streams. As for utilities, all assigned streams and exchanges are flagged.

Upon completing the above procedure for single exchanger streams, the algori moves on to multi exchanger streams. When assigning these streams the same te niques as for the single exchanger assigning applies. In order to properly sort and k track on possible exchanges, the module makes use of a support module called H Gen. This is a module that is run every time an assignment of multi exchanger strea is done, and the module is therefore relatively short to avoid excessive time utilizati Once feasible exchanges are discovered, only low quality heat is used when assigni In contrast with the utility and single exchanger assigning, this procedure is only p formed for hot multi exchanger streams. This due to the fact that cold multi exchan streams are covered through the cold single stream assigning as well as the hot mu stream assigning. Flagging is performed when assigning these exchanges as well.

As can be seen from figure 5.1, the single and multi exchanger assigning is perform in loops. As new feasible exchanges may arise when assigning streams in the abo manor, the procedures are done several times in order to assign as many exchanges possible. By setting the loop count very high it is possible to ensure that all feasib exchanges are assigned. By forcing the algorithm to perform operations only f unflagged streams with feasible exchanges present, while all other streams a exchanges are ignored, the high loop count does not lead to excessive run times. Cu rently the module loops a total of 50 times with a run time of approximately two se onds.

Upon the final loop completion, the algorithm may or may not have achieved a fu feasible starting solution. This solution is in case an entirely serial configuration, a will therefore potentially have lower piping cost than its parallel opposite, as well increased controllability as previously mentioned. However, in many cases there w still be remaining exchanges that cannot be assigned using the methods describ above. The algorithm therefore tries to achieve a parallel split for these remaini streams. In contrast with methods described in the serial/parallel mCp generator, t part of the algorithm uses a more precise method for setting the split. While the m simple serial/parallel generator used only heat loads to set splits, this module uses h loads and temperature differences in order to obtain a more reasonable split.

In order to properly explain the procedure, an example will be used. Given exchan between streams H1 and C1 as well as between H1 and C2, where both exchanges infeasible given the methods designing serial structures. However, the exchanges possible when using a split. Stream H1 is the stream that requires the split, while

cold streams apparently have a serial configuration as they are single exchanger structures.

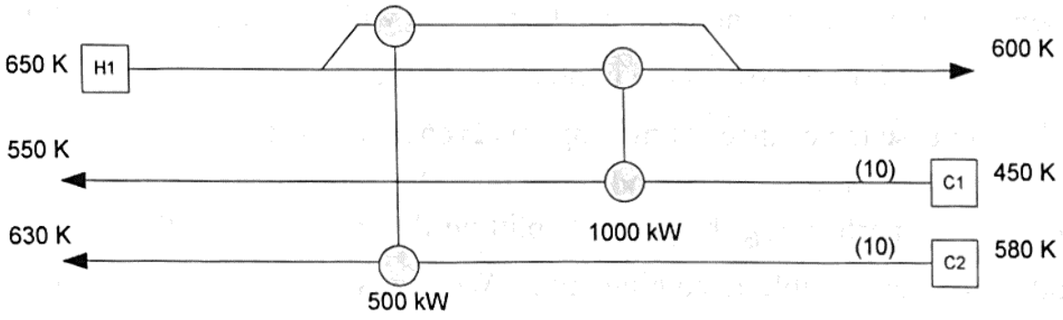

Figure 5.3 - Stream split

With heat loads of 1000 kW and 500 kW respectively, and temperature differences between inbound hot stream and outbound cold stream of 100 and 20 Kelvin, as well as mass flow rates of 10 for both cold streams, an intermediate mathematical term called paraT is calculated for each exchange. The term is calculated by taking the temperature difference described above and adding heat load divided by mass flow for the stream that is not to be split. In our example the calculations would be 100 + 1000/10 = 200 and 20 + 500/10 = 70. These numbers are fed into a new calculation giving a new intermediate term called alfakeys. These terms are in fact just heat loads divided by the paraT term calculated above. In our example the paraTs will be equal to 1000/200 = 5 and 500/70 = 7,14. These numbers form the basis for a split key that gives the percentage of flow in each pipe. The calculation is simple; through the first exchanger a flow of 5 / (5 + 7,14) = 0,41 (or 41%) and 7,14 / (5 + 7,14) = 0,59 (or 59%). H1 should therefore have a split sending 41% of its mCp to the exchange with C1 while 59% should be sent to the exchange with C2.

Table 5.1 - Split calculation

Exchange: H1 - C1		Exchange: H1 - C2	
ParaT	200	ParaT	70
AlfaKey	5	Alfakey	7,14
SplitKey	0,41	SplitKey	0,59

When using just heat loads the routing would be 67% to the first exchanger and 33% to the second. To see why the 41/59 split is more reasonable then the more primitive variant from the serial/parallel mCp generator, one can examine the temperature differences. There is a much greater difference in temperature present at the first exchange

than at the second. The degree of freedom is therefore larger at the first exchange than at the second. In other words the exchange is tighter between H1 and C2 than between H1 and C1. When considering this it is obvious that sending a larger mass flow to the exchange between H1 and C1 in many cases will render the exchange between H1 and C2 infeasible. As a greater degree of freedom is present in the first exchange, the mass flow sent to this part of the network can be reduced without the same risk of infeasibility. The increased mass flow to the tighter part of the network will in most cases considerably increase the chances of making this exchange feasible.

Before actually performing the parallel splitting described above, the module checks feasibility for the possible remaining splits. When assigning hot parallel splits as the one described above, splitting is only performed if the exchanges are in fact feasible as splits. If feasible splits are not possible, the algorithm skips the allocation.

The module also tries to assign possible cold splits as a final step. Currently this part of the algorithm is done without the feasibility check. The reason for handling the splits differently is the desire to have a consistent mass flow. It is better to assign infeasible exchanges in this final step than not assigning them at all. This is mainly due to the need for consistent mass flow as input data in the modules initializing all other variables. These other modules are further described later in this chapter.

Further tests will show that this algorithm provides feasible starting solutions for numerous problems and provides near feasible solutions for others. More details on these tests can be found in chapter eight. Notice that in later chapters, this module is referred to as mode 5, while the serial/parallel and clever modules are referred to as mode 1-3 and mode 4 respectively. The change in notation is due to reporting convenience.

5.5 Other visual basic modules

As mentioned introductory, the mCp variables form the basis for all other variables in the HENS problem. Using the information created by the various mCp generators, it is possible to calculate temperatures in all arcs as well as other variable values.

Initially, several small modules utilized the information gathered from the mCp generators. One module calculated temperatures in all arcs by using the mass flow combined with the heat loads and heat nodes. The processed values were all fed back to the sheet mCpgen and formed the input data for three other small modules.