DHP
Given directed graph (V, E) and two nodes x, y, decide whether there exists a Hamiltonian path from x to y.

Theorem 5.14.
HP is polynomially reducible to DHP, thus DHP is NP-complete, too.

Proof. Replace an undirected graph by a directed one with two directed edges (with opposite directions) for every undirected edge of the undirected graph. □

As a final NP-complete problem that we use in a bioinformatics reduction we treat MAX-CUT which is defined as follows: given an undirected graph (V, E) and an assignment of non-negative numbers to its edges, partition nodes into two disjoint and non-empty sets A and B such that the sum of numbers assigned to edges between nodes in A and nodes in B is a big as possible.

MAX-CUT
Given undirected graph (V, E) with weights assigned to edges, and a lower bound b, decide whether there is a partition of V into subsets A, B with at least b edges between nodes of A and nodes of B.

NP-completeness of a variant of 3SAT that is called 3NAESAT (with 'NAE' standing for "not all equal") is required to show NP-completeness of MAX-CUT. Given a 3SAT-formula, 3NAESAT asks for a truth-value assignment that makes at least one, but not all of the literals within every clause 'true'. Its NP-completeness is obtained from satisfiability problem for Boolean circuits. For details we refer the reader to [61] (Theorem 9.3). To give an impression to the reader of what 'NAE-problem' are we treat here the considerably simpler 4NAESAT problem.

4NAESAT
Given a Boolean formula in 4-clausal form, is there a truth-value assignment that makes at least one literal per clause 'true', and at least one literal per clause 'false'.

Theorem 5.15.
3SAT is polynomially reducible to 4NAESAT, thus 4NAESAT is NP-complete, too.

Proof. Let an instance $\varphi = (L_{11} \vee L_{12} \vee L_{13}) \wedge \ldots \wedge (L_{n1} \vee L_{n2} \vee L_{n3})$ of 3SAT be given. Expand it to an instance of 4NAESAT by introducing in each clause the same new Boolean variable z obtaining formula

$$\psi = (L_{11} \vee L_{12} \vee L_{13} \vee z) \wedge \ldots \wedge (L_{n1} \vee L_{n2} \vee L_{n3} \vee z) \; .$$

Assume that truth-value assignment $\Im$ satisfies φ. Setting variable z to 'false' defines a truth-value assignment which satisfies between 1 and 3 literals within each clause of ψ . Conversely, let $\Im$ be a truth-value assignment that satisfies between 1 and 3 literals within each clause of ψ. Then the modified truth-value assignment $\neg\Im$ with $\neg\Im(x)$ = 'true' if $\Im(x)$ = 'false', and $\neg\Im(x)$ = 'false' if $\Im(x)$ = 'true' also satisfies between 1 and 3 literals per clause of ψ. Choose among $\Im$ and $\neg\Im$ the truth-value assignment which assigns 'false' to z. Obviously, this truth-value assignment satisfies every clause of φ. □

Theorem 5.16.
3NAESAT is polynomially reducible to MAX-CUT, thus MAX-CUT is NP-complete, too.

Proof. Let a 3SAT-formula

$$\varphi = (L_{11} \vee L_{12} \vee L_{13}) \wedge \ldots \wedge (L_{n1} \vee L_{n2} \vee L_{n3})$$

containing Boolean variables $x_1, \ldots, x_m$ be given. Let a_j be the number of occurrences of x_j in φ, b_j the number of occurrences of $\neg x_j$ in φ, and n_j be $a_j + b_j$. Thus, $n_1 + \ldots + n_m = 3n$. We construct a graph having $2m$ nodes called x_j and $\neg x_j$, for $j = 1, \ldots, m$. Weighted edges are introduced as shown in Fig. 5.11 with three edges weighted 1 for each of the clauses, and a single edge weighted $2n_j$ for each pair of literals x_j and $\neg x_j$. In case that some clause contains only 2 different literals its corresponding triangle degenerates to two nodes and a single edge weighted 2 instead of 1 for proper triangle edges. We are looking for a cut of size at least $8n$.

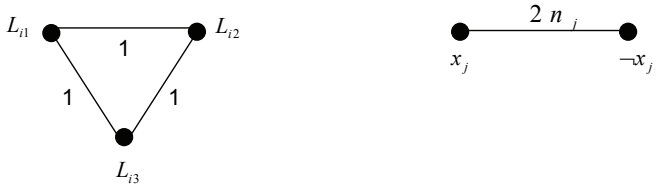

Fig. 5.11. Reducing 3NAESET to MAX-CUT

Lemma 5.17.
Assume that truth-value assignment $\Im$ satisfies 1 or 2 literals within each clause. Then the weighted graph constructed above admits a cut of size at least $8n$.

Proof. Put into node set A of the cut all nodes that correspond to literals that are evaluated 'true' by $\Im$, and put into node set B of the cut all nodes that

correspond to literals that are evaluated 'false' by $\Im$. Thus, non-triangle edges between any two pairs of nodes x_j and $\neg x_j$ contribute $2(n_1 + \ldots + n_m) = 6n$ to the size of the cut. Since every triangle representing a clause has nodes in both parts of the cut, its edges contribute exactly 2 to the size of the cut. Thus, all triangle edges together contribute exactly $2n$ to the size of the cut. All together, size of the cut is exactly $8n$. □

Lemma 5.18.
Assume that the weighted graph constructed above admits a cut of size at least $8n$. Then there exists a truth-value assignment $\Im$ that satisfies 1 or 2 literals within each clause of formula φ.

Proof. Consider a cut of size at least $8n$. If there is a variable x_j such that both nodes corresponding to x_j and $\neg x_j$ are within the same part of the cut, then triangle edges contribute at most $2a_j + 2b_j = 2n_j$ to the size of the cut. By putting nodes corresponding to x_j and $\neg x_j$ into different parts of the cut we would gain $2n_j$ in size, but loose at most $2n_j$ in size, thus obtain a new cut with size that is still at least $8n$. Thus we may assume that for each variable x_j, nodes corresponding to x_j and $\neg x_j$ occur in different parts of the cut. Taken together, non-triangle edges between these nodes contribute exactly $6n$ to the size of the cut. Thus at least a value of $2n$ must be contributed to the size of the cut by triangle edges. This only happens if each triangle has nodes in both parts of the cut. Thus we have exactly the situation as in the proof of the lemma before. Defining truth-value assignment $\Im$ in such a way that exactly the literals corresponding to nodes in one part of the cut are set 'true' we enforce that in every clause 1 or 3 literals are set 'true' by $\Im$. □

With Lemma 5.17 and 5.18 the proof of Theorem 5.16 is complete. □

5.4 Bridge to Bioinformatics: Shortest Common Super-Sequence

As an intermediate problem on the way towards NP-completeness of the sum-of-pairs multiple alignment problem MA, the shortest common super-sequence problem SSSEQ (see [53]) is shown to be NP-complete. This is done by reducing VERTEX COVER to SSSEQ. Later SSSEQ is further reduced to MA.

We fix some notations. We call string $T = T[1 \ldots n]$ a *super-sequence* for string $S = S[1 \ldots m]$ if the characters of S can be embedded under preservation of ordering into T, that is if there are indices $i_1, \ldots, i_m$ such that $1 \leq i_1 < \ldots < i_m \leq n$ and $S(1) = T(i_1), \ldots, S(m) = T(i_m)$. Thus the characters of S occur in a super-sequence T in the same order as they occur in S, though they need not occur contiguously in the super-sequence T. SSSEQ is the problem of finding a shortest common super-sequence for given strings $S_1, \ldots, S_k$. Formulated as a decision problem, SSSEQ asks whether a common super-sequence T of length at most b exists, given strings $S_1, \ldots, S_k$ and upper bound b.