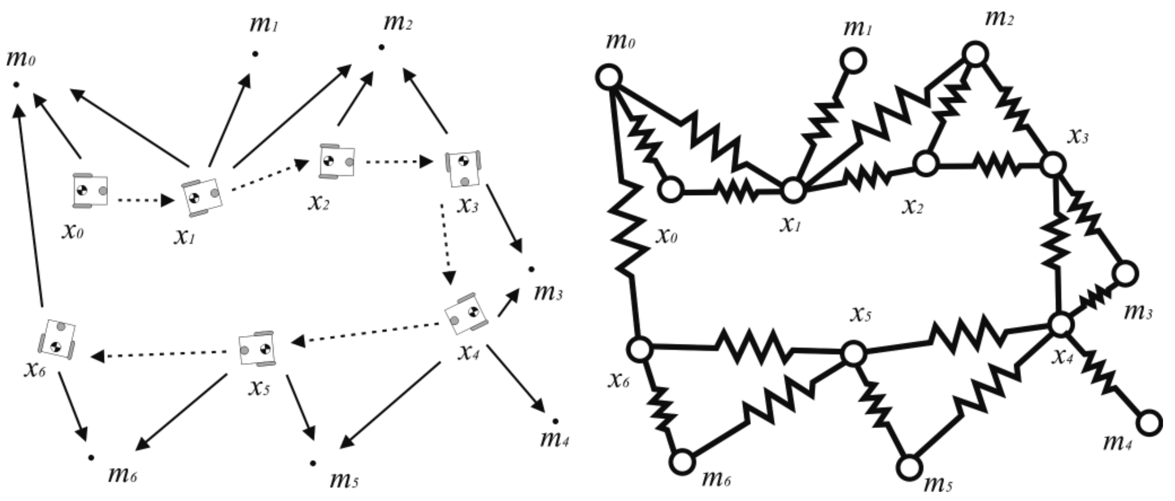

Figure 5.47 Evocative illustration of the graph construction. Constraints between nodes in the graph are represented as "soft" constraints (like springs). The solution of the SLAM problem can then be computed as the configuration with minimal energy.

odometry input u_t) and the relative position between the robot locations and the features observed from those locations.

The key property to remember about graph-based SLAM is that the constraints are not to be thought as rigid constraints but as soft constraints (figure 5.47). It is by relaxing these constraints that we can compute the solution to the full SLAM problem, that is, the best estimate of the robot path and the environment map. In other words, graph-based SLAM represents robot locations and features as the nodes of an elastic net. The SLAM solution can then be found by computing the state of minimal energy of this net [139]. Common optimization techniques to find the solution are based on gradient descent, and similar ones. A very efficient minimization procedure, along with open source code, was proposed in [141].

There is a significant advantage of graph-based SLAM techniques over EKF SLAM. As we have seen, in EKF SLAM the amount of computation and memory requirement to update and store the covariance matrix grows quadratically in the number of features. Conversely, in graph-based SLAM the update time of the graph is constant and the required memory is linear in the number of features. However, the final graph optimization can become computationally costly if the robot path is long. Nevertheless, graph-based SLAM algorithms have shown impressing and very successful results with even hundred million features [79, 116, 117, 173, 315]. However, these algorithms attempt to optimize over the entire robot path and were therefore implemented to work offline. Some of the online implementations used submap approaches.