15 Simulated Trading Applied to TSP

15.1 Application of Simulated Trading to the TSP

Simulated trading (ST) was originally developed for the vehicle routing problem (VRP), which is an extension of the traveling salesman problem (TSP), as already mentioned in Chap. 2. In their original publication [13], Bachem et al. considered each truck driver as an agent who buys and sells goods to his customers.

In contrast, there is only one traveling salesman in a TSP. However, the ST algorithm can also be applied to the TSP by splitting the roundtrip such that several agents $1, \ldots, M$ each get a partial sequence $\sigma(k_i), \sigma(k_i+1), \ldots, \sigma(k_i+N_i-1)$, containing N_i nodes, of the whole tour. Of course, all these sequences put together must result in the closed roundtrip of the traveling salesman, i. e., $\sum_{i=1}^{M} N_i = N$.

Now, the ST algorithm requires that there be items to be bought or sold and that there be agents willing to give a buy or a sell order. In this section, we want to stick closely to the original approach as described in [13]. The individual items in this case are the nodes of the tour, which are sold and bought by agents, thus removing or adding one node from or to their partial sequences.

The outline of such a "trading move" is as follows:

- First, it is determined whether an agent wants to buy or to sell something. For simplicity, only the case where each agent buys or sells one node will be considered. Thus, optimally there will be as many buying as selling agents.
- Then each selling agent i must submit a sell order:
  - He/She calculates for each of the nodes $\sigma(k_i+1), \ldots, \sigma(k_i+N_i-2)$ the savings $S(j,i)$ that would occur if the jth node in the tour was removed, i. e.,

    $$S(j,i) = \Delta_j^- \,, \tag{15.1}$$

    with

    $$\Delta_j^- = D(\sigma(j-1),\sigma(j)) + D(\sigma(j),\sigma(j+1)) - D(\sigma(j-1),\sigma(j+1)) \,. \tag{15.2}$$

    Note that we do not consider the first and last nodes of such a partial sequence as they are connected with the outer nodes of the partial sequences of the neighboring agents and as the agents are considered to act independently of each other.

  - Now the question of the behavior or manners of each agent arises. If they are rather greedy, they would want to offer that node $\sigma(j)$ for which $S(j,i)$ is maximal. However, as was already mentioned in [13], it is advantageous to work with a nongreedy behavior. Therefore, probabilities $p(j)$ are assigned to the nodes $\sigma(j)$

$$p(j) = \frac{S(j,i)}{\sum\limits_{l=k_i+1}^{k_i+N_i-2} S(l,i)} \tag{15.3}$$

    for all nodes $\sigma(j)$ with $k_i+1 \leq j \leq k_i+N_i-2$. Note that the probability of a particular j is large if the corresponding savings $S(j,i)$ is large compared to the other savings values.
  - Then a uniformly distributed random number r is chosen by which a particular $\tilde{j}$ is determined according to the probabilities $p(j)$. Thus, agent i offers the node $\sigma(\tilde{j})$, for which he/she gets the savings $S(\tilde{j},i)$, for sale. Let us introduce the abbreviation $S(\sigma(\tilde{j})) := S(\tilde{j},i)$ here, thus assigning the savings value to node $\sigma(\tilde{j})$.
- All selling agents send their sell orders with the associated savings values to the central stock market, which summarizes them in a public selling list. This selling list, containing Z offers, is sent to all buying agents.
- Each buying agent must then go through each individual offer:
  - For each offered node z, an agent i has to find out where to best insert z into his/her partial sequence. So here a greedy approach is used. Let $C(z,i)$ denote the minimum costs for inserting node z into the partial sequence of agent i:

$$C(z,i) = \min_{j=k_i,\ldots,k_i+N_i-1} \left\{\Delta^+_{z,j}\right\} , \tag{15.4}$$

    with

$$\Delta^+_{z,j} = D(\sigma(j),z) + D(z,\sigma(j+1)) - D(\sigma(j),\sigma(j+1)) \,. \tag{15.5}$$

  - Then agent i must determine which node $\tilde{z}$ he/she actually wants to buy. If only the costs for insertion are considered, then nearly every agent would refuse to buy, as the costs for insertion are nonnegative and only zero if node z lies on a straight line between $\sigma(j)$ and $\sigma(j+1)$. Therefore, the gain for removing a node in the old sequence and inserting it into the new one, i. e.,

$$G(z,i) = S(z) - C(z,i) \,, \tag{15.6}$$

    must be considered. If this gain is positive, then the shifting of the node to the new agent decreases the tour length. Now agent i determines the minimum gain m_i,

$$m_i = \min_{z \in \text{ selling list}} G(z,i)\,. \tag{15.7}$$

If $m_i < 0$, then there is at least one offer with negative gain. Of course, the single agent i would not want to accept such an offer. Whats more, if all gains are negative, he would not want to buy anything at all. On the other hand, it might be advantageous for the system as a whole if he accepted an offer with negative gain so that the overall system would not get stuck in local minima. Therefore, the algorithm distinguishes between the cases $m_i < 0$ and $m_i \geq 0$:

- If $m_i \geq 0$, then all of the gains of an agent are nonnegative, so that he can be happy with any offer. Analogously to the behavior of the selling agents, the buying agent i calculates the probabilities

  $$p(z,i) = \frac{G(z,i) - m_i}{\sum\limits_{x=1}^{Z} (G(x,i) - m_i)} \tag{15.8}$$

  for the various offers z. However, checking this formula closely, one finds that the least attractive offer has a probability of 0. To overcome this problem, the value m_i is slightly decreased, e. g., set to $0.97 \times m_i$, before the probabilities are calculated. Then a random number is calculated by which an offer $\tilde{z}$ that agent i would accept is determined. Thus, agent i tells the stock market that he wants to submit a buy order for node $\tilde{z}$ with a gain $G(\tilde{z}, i)$.
- Otherwise, in the case $m_i < 0$, a trick is used: agent i adds the virtual $(Z+1)$th offer $\hat{z}$ with the supposed gain $G(\hat{z}, i) = 0$ to the list of offers. Analogously to the case above, m_i is decreased by 3% by multiplying it by 1.03, such that the least attractive offer also has some probability of being chosen. The probabilities $p(z_1, i)$, $p(z_2, i)$, …, $p(z_Z, i)$, and $p(\hat{z}, i)$ for the individual offers are calculated as follows:

  $$p(z,i) = \frac{G(z,i) - m_i}{\sum\limits_{x=1}^{Z+1} (G(x,i) - m_i)}\,. \tag{15.9}$$

  By means of a uniformly distributed random number, a particular offer $\tilde{z}$ is chosen from the list of offers. If $\tilde{z} \neq \hat{z}$, then agent i tells the stock market that he/she wants to buy node $\tilde{z}$ with the determined gain $G(\tilde{z}) = G(\tilde{z}, i)$. If, however, $\tilde{z} = \hat{z}$, then he/she tells the stock market that he/she does not want to buy anything.

- The stock market collects all these buy orders from the buying agents in a list. Comparing this list with the selling list, there are three possibilities for each node on the selling list:

  - There might be no buy order for a node on the selling list. Then the agent who made this sell order cannot sell his/her node. Therefore, this sell order is removed from the selling list.
  - There is exactly one buy order for a node on the selling list. In this case, the stock market creates a link between the buy order and the corresponding sell order. The link is weighted with the gain that would be received for shifting the node from the selling agent to the buying agent.
  - There might be more than one buy order for a node on the selling list. In this case, the stock market goes through all of these buy orders, determines that buy order with the maximum gain, and creates a link between this buy order and the corresponding sell order. This link is weighted with the gain.
- Finally, the stock market sums up the gains over all existing links. If the sum over all gains is nonnegative, then the trading move is accepted, i. e., for all existing links between a sell order from a selling agent and a buy order from a buying agent, the node is shifted from the selling to the buying agent. Note that some of the partial gains might be negative, producing local deteriorations. However, if the sum of all gains is positive, this trading move leads to an overall improvement.

If the trading move is always performed in this way, then it always exhibits in some sense the same size, as each selling agent gives a sell order. In order to add even more randomness and to improve the results, the authors introduced a probability q with which a selling agent actually places a sell order [13].

This trading move is the key ingredient of the ST algorithm. Thus, the outline of the ST algorithm is as follows:

- Read the data, determine the distance matrix, and initialize the system with a random tour.
- Do several times:
  - Split the tour into an even number of partial sequences. These sequences should be roughly the same length. Each of these sequences must contain at least two nodes. Furthermore, do this splitting in such a way that a tour position is selected randomly that is the starting position of the partial sequence of the first agent.
  - Do several times:
    - Determine which agents want to buy and which ones want to sell. The number of buying and selling agents should be roughly the same.
    - The individual selling agents determine by means of a random number whether they actually want to sell. This random number must be smaller than the probability q mentioned above. If it is, then the agents check whether there are at least three nodes in their partial sequences of the tour and determine the node they want to sell together with the savings for removing this node from the partial sequence.
    - The stock market collects all sell orders from the selling agents and summarizes them in a public selling list.

    - If the selling list contains at least one item, then it is sent to the buying agents.
    - The buying agents go through all offers and determine for each offered node where to best insert it into their partial sequences. Then they choose one of the offers randomly as described above and send a buy order for this offer together with its gain to the central stock market.
    - The stock market goes through all buy orders and creates links between the sell orders and those buy orders that provide the maximum gain for the corresponding sell orders. The links are weighted with the determined gains.
    - If the sum over all these gains is nonnegative, then this trading-move is accepted, i. e., for each link between a selling or a buying agent, the offered node is removed from the partial sequence of the selling agent and inserted at the best possible position into the partial sequence of the buying agent. Otherwise, if the move is not accepted, nothing happens.
  - After several such trading moves, the partial sequences are merged to one closed tour such that it can next be split into other parts.
- Print out the final result.

15.2 Computational Results

We performed several simulations with the algorithm sketched above. We set the number of buying agents equal to the number M_s of selling agents such that M_s is given by $M/2$, with M being the overall number of agents. We give each agent roughly the same number of nodes when splitting the tour into several partial sequences, namely, $[N/M]$ or $[N/M]+1$, with $[x]$ being the integer part of x, such that the sequences merged together result in a feasible configuration containing each node once. Furthermore, we add the first node of each partial sequence to the end of the previous sequence such that there are no connections of the tour that must not be cut. Thus, $\sum_{i=1}^{M} N_i = N + M$.

For us, the two main variables of interest are the number $M_\mathrm{s} = M/2$ of selling agents and the probability q with which a selling agent actually wants to sell something. Figure 15.1 shows the mean results (with error bars) for three TSP instances achieved with the ST heuristic. In each simulation, 10000 sweeps were performed, which is according to our empirical tests enough to "freeze" the system within this heuristic. In each sweep, first the tour was split among the agents, giving the individual agents partial sequences as described above. Then N trading moves were tried. At the end of each sweep, the partial sequences were merged together to form a closed roundtrip through all nodes again. Note that the results shown generally in this chapter, and thus also in Fig. 15.1, are averaged over only 50 optimization runs each.

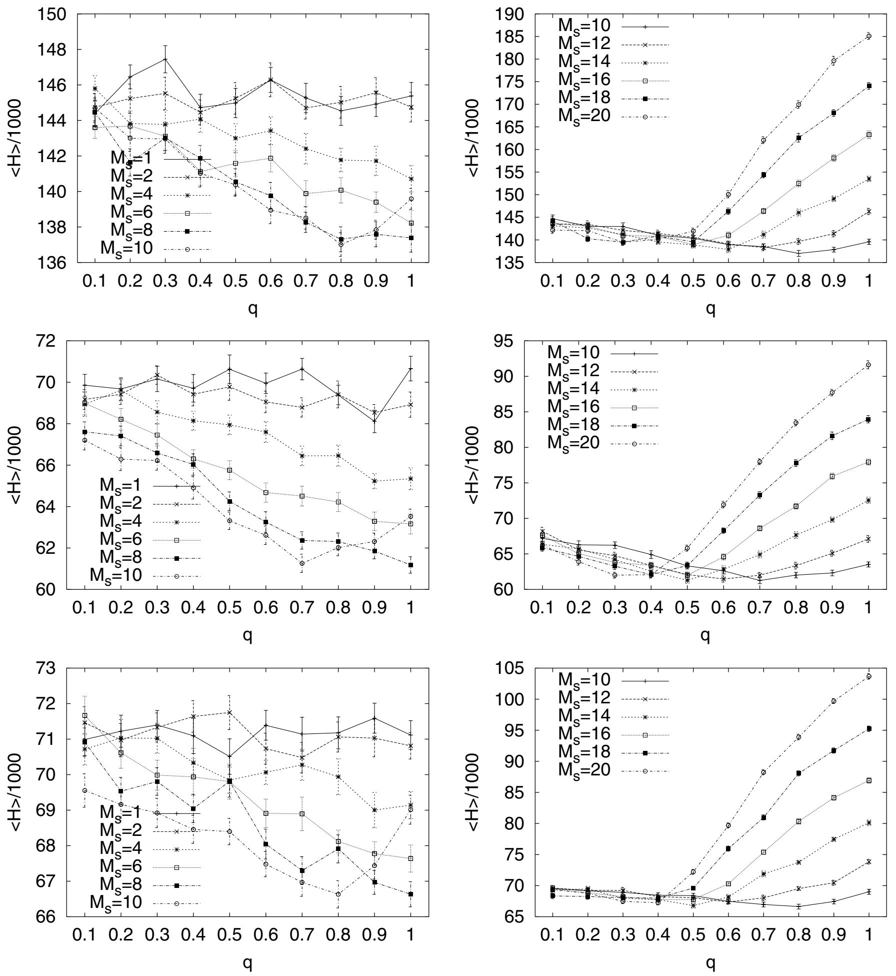

Fig. 15.1. Quality of the results achieved with the ST heuristic for three TSP instances [BEER127 (*top*), LIN318 (*middle*), and PCB442 (*bottom*)] depending on the propability q and on the number M_s of selling agents (*left*: results for $M_\mathrm{s} \leq 10$, *right*: results for $M_\mathrm{s} \geq 10$)

We find that the results must be differentiated between a small number of agents and a large number of agents. This splitting can be done at $M_\mathrm{s} = 10$ for all three instances: for $M_\mathrm{s} < 10$, the results improve with increasing M_s and increasing q. For all three instances, the curve for $M_\mathrm{s} = 10$ starts to rise again at $q \to 1$. This effect becomes dramatic for $M_\mathrm{s} > 10$: with increasing q, the results become worse. The more agents are used, the worse the results become. Obviously, this is the reason for introducing the parameter q into the algorithm, namely, to obtain much better results for larger numbers of agents. It seems that there is an optimum number of agents taking part in such a trading move: if the number of agents is too small, then the number of sell offers is small, so that there is no real stock market. The trading move simply performs one or a few node insertion moves (NIMs) at the same time. Thus, the trading move is identical to either the NIM or to some special cases of the Lin-n-opts with small n. These moves are accepted if they either improve the configuration or do not worsen it too much. If the number of sell offers is large, the trading move, which creates a large amount of links between sell and buy offers and thus changes the system overall in many ways, is no longer a move of the local search type. As already discussed in Chap. 5, the more cuts are introduced in a Lin-n-opt, the less probable the move will lead to an improvement if the current configuration is already rather good. This finding for the Lin-n-Opts that $n \leq 3$ is optimal is found here again for the trading move that leads to the best results if the number of selling agents taking part in the trading move is roughly 7 or 8. The shift in the optimum number of cuts is given by the various acceptance rules, which are rather elaborate for the trading move.

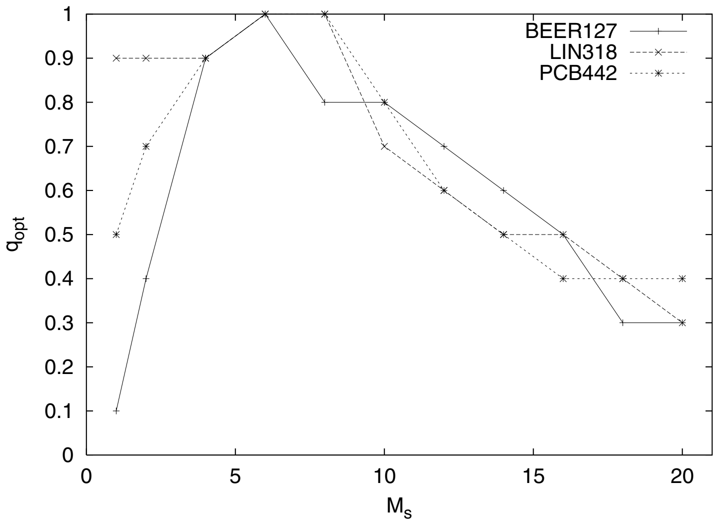

Fig. 15.2. Optimum probability leading to the best results as a function of M_s, the number of selling nodes, for three TSP instances

Figure 15.2 shows the optimum value q_opt found for the probability q for various numbers of selling agents. Please note that the results at small M_s are not significant as there the error bars overlap with each other. But basically we again obtain the picture that an introduction of the probability q with values $q \leq 0.4$ is necessary for a large number of agents in order to decrease the number of agents taking part in a trading move and thus to increase the probability that the trading move will lead to a good result.

15.3 Discussion of Simulated Trading

The ST heuristic is at first glance a very elaborate technique for getting solutions to combinatorial optimization problems. The approach of ST seems to be well balanced between a strict set of rules and some kind of randomness, which is necessary to arrive at quite good results with a stochastic optimization algorithm.

However, as the results in the last section show, this heuristic is not well suited for the TSP. (Thus, we did not consider even applying it to larger instances.) In fact, the results are at least more than 10% worse than the optimum value. Of course, we must admit that this algorithm was originally not developed for the TSP and that the artificial splitting of the tour into partial sequences leads to a localization of the problem. A complex problem like the TSP, however, is defined by long-range interactions of its individual parts. One must look at the complete problem in a global way. Thus, ST can surely not compete with global optimization algorithms like simulated annealing for the TSP. It may be better suited for VRP and depot localization problems, for which it was designed and for which such a splitting into the subsets of customers served by one truck or by one depot even makes sense.

But if the ST approach wants to make use of the local search concept, then the number of agents taking part in a trading move must not be too large. For a small number of them, the trading move is obviously applied in a greedylike way, i. e., it mostly leads to improvements, until no further improvements can be found due to the limitations in the conception of the trading move. With an increasing number of orders, good moves can be found even more seldom.

15.4 Simulated Trading and Working

According to Benedict of Nursia's *Ora et labora*, a man should pray and work. At least getting agents to work is easy. In the original ST approach, the agents only trade but do not work, e. g., to improve their parts of the tour. Thus, we extend this original approach now, so that after each trading move all agents whose tours contain at least $N_i = 4$ nodes perform N_i moves in the greedy mode. The moves are chosen randomly to be either an exchange or a node insertion move or a Lin-2-opt with equal probability. Larger moves were not used in our experiments as the partial sequences are rather short for a larger number of agents.

Figure 15.3 shows the results achieved with this simulated trading and working (STW) approach. We find that the results for a small number of agents improve to a large extent: the best results are achieved for $M_\mathrm{s} = 1$, followed by $M_\mathrm{s} = 2$ and $M_\mathrm{s} = 4$. Here the number of agents is minimal, meaning their partial sequences are rather long. Furthermore, we find that the results for $M_\mathrm{s} = 1$ and $M_\mathrm{s} = 2$ are rather independent of q, in contrast

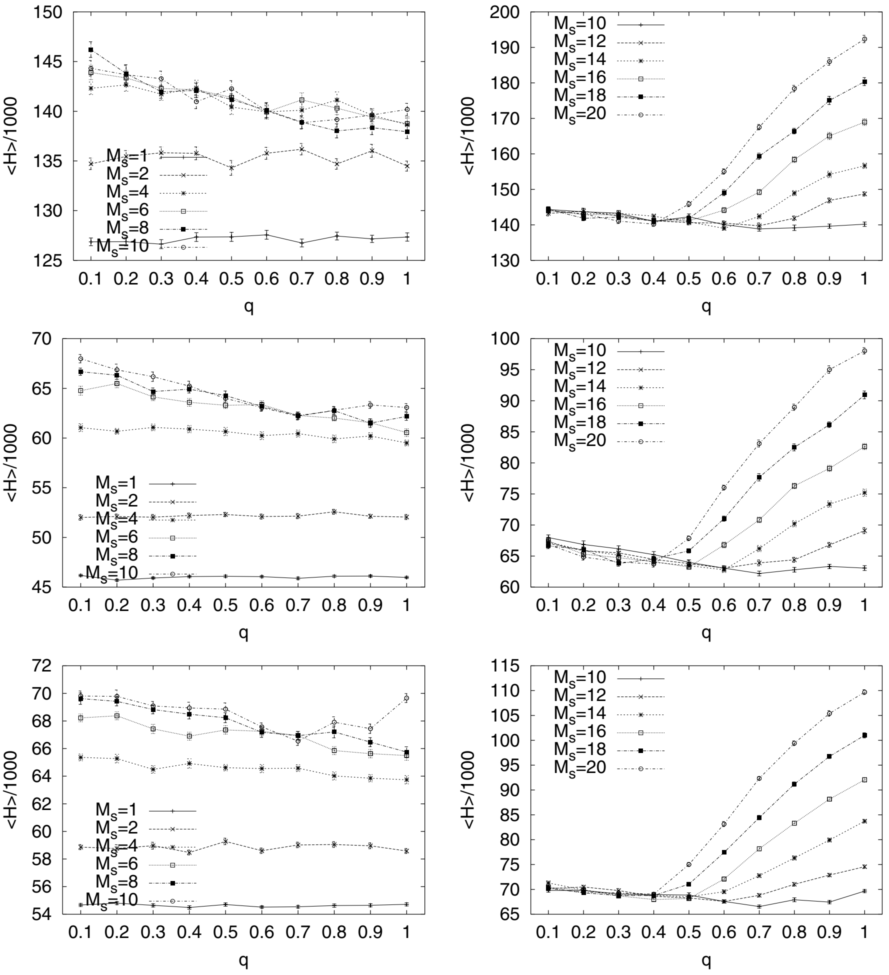

Fig. 15.3. Quality of the results achieved with the STW heuristic for three TSP instances [BEER127 (*top*), LIN318 (*middle*), and PCB442 (*bottom*)] depending on the propability q and on the number M_s of selling agents (*left*: results for $M_\mathrm{s} \leq 10$, *right*: results for $M_\mathrm{s} \geq 10$)

to the results for larger M_s. As for these small M_s, the largest improvements were found in comparison with the pure ST approach. We conclude that these improvements are mainly due to the small moves performed in the greedy mode. Of course, here the greedy algorithm must be most successful as the partial sequences are rather long for small M_s. For $M_\mathrm{s} \geq 10$, we again find that the results become worse with increasing probability q. Thus, we have the same dependency on q and M_s here as before for large M_s. However, the results are worse than those for the original ST approach for large M_s and large q, although the greedy moves should be able to optimize short partial sequences very well. But obviously the trading moves not only destroy the local order generated by the greedy moves but are even less able than before to lead to reasonable configurations. This is due to the fact that with preoptimized partial sequences, the gain for removing a node is mostly much smaller than the costs for inserting it into another partial sequence. Thus, a good trading move can hardly be found anymore. One could even say that the local greedy moves work against the trading moves.

Summarizing, we get in this STW approach better results for small M_{s}, where we basically benefit from the greedy algorithm when it is applied with the small moves but worse results for large M_{s} than for the original ST approach. In all cases, however, the quality of the results is not satisfactory. Thus, we recommend that the agents start to pray.