

CHAPTER 47

Agent-Based Traffic Assignment

Kai Nagel and Gunnar Flötteröd

47.1 Introduction

This chapter presents MATSim from a DTA perspective. The following material is an abridged and edited version of Nagel and Flötteröd (2012).

The traffic assignment problem, whether macroscopic or microscopic, static or dynamic, trip-based or agent-based, is to identify a situation where travel demand and travel supply (network conditions) are consistent with each other. Travel demand results from a demand model that reacts to conditions in the network; these are the output of a supply model (network loading model) using travel demand as its input. A solution of the traffic assignment problem describes an equilibrium between travel demand and travel supply.

Possibly, the most intuitive mathematical formulation of this problem is defined by a fixed point: Find a demand pattern generating network conditions that, in turn, cause the same demand pattern to re-appear. This formulation is operationally important because it motivates a straightforward way of calculating an equilibrium by alternately evaluating the demand model and the supply model. If these iterations stabilize, a fixed point is attained that solves the traffic assignment problem.

The remainder of this chapter places MATSim into this DTA framework. Section 47.2 starts out from the static and macroscopic assignment of route flows and incrementally enriches this formulation into a dynamic and fully disaggregate agent-based assignment problem. Section 47.3 then turns to the problem of how to simulate (solve) this model system, with a particular focus on MATSim's coevolutionary approach. Section 47.4 concludes the presentation.

How to cite this book chapter:

Nagel, K and Flötteröd, G. 2016. Agent-Based Traffic Assignment. In: Horni, A, Nagel, K and Axhausen, K W. (eds.) *The Multi-Agent Transport Simulation MATSim*, Pp. 315–326. London: Ubiquity Press. DOI: <http://dx.doi.org/10.5334/baw.47>. License: CC-BY 4.0

47.2 From Route Swapping to Agent Plan Choice

The following details an increasingly comprehensive specification of the traffic assignment problem, starting from the classical static user equilibrium model and ending with a fully dynamic model that captures arbitrary travel demand dimensions at the individual level. Computationally, the iterative fixed point solution procedure is carried throughout the entire development. Deliberately, this solution method also has a behavioral interpretation as a model of day-to-day replanning; see also Section 97.3.5.

We start by considering route assignment only. The generalization towards further choice dimensions will turn out to be rather straightforward.

47.2.1 Static Traffic Assignment

Consider a network of nodes and links, where some, or all, of the nodes are demand origins, denoted by o , and/or demand destinations, denoted by d . The constant demand q^{od} in an O-D relation od splits up among a set of routes K^{od} . Denote the flow on route $k \in K^{od}$ by r_k^{od} , where $\sum_{k \in K^{od}} r_k^{od} = q^{od}$.

Most route assignment models either specify a UE (User Equilibrium a.k.a. Wardrop's first principle) or an SUE (Stochastic User Equilibrium). A UE postulates that r_k^{od} is zero for every route k of non-minimal cost (Wardrop, 1952):

$$c(k) = \min_{s \in K^{od}} c(s) \Rightarrow r_k^{od} \geq 0 \quad (47.1)$$

$$c(k) > \min_{s \in K^{od}} c(s) \Rightarrow r_k^{od} = 0 \quad (47.2)$$

where $c(k)$ is the cost (typically delay) on route k .

An alternative, frequently-used approach is to distribute the demand onto the routes such that an SUE is achieved, where users have different perceptions of route cost and every user takes the route of *perceived* minimal cost (Daganzo and Sheffi, 1977). Mathematically, this means that the route flows fulfill some distribution

$$r_k^{od} = P_k^{od}(c(x(\{r_k^{od}\}))) \cdot q^{od} \quad (47.3)$$

where the route splits P_k^{od} are a function of the network costs $c(x)$, which depend on the network conditions x , which, in turn, depend on all route flows $\{r_k^{od}\}$.

In either case, the model needs to be solved iteratively, which typically involves the following steps (Sheffi, 1985):

Algorithm 47.1 Macroscopic and static route assignment

1. **Initial conditions:** Compute some initial routes (e.g., best path on empty network for every O-D pair).
 2. **Iterations:** Repeat the following many times.
 - (a) **Network loading:** Load the demand on the network along its routes and obtain network delays (congestion).
 - (b) **Choice set generation:** Compute new routes based on the network delays.
 - (c) **Choice:** Distribute the demand between the routes based on the network delays.
-

Defining the network loading as more on the “physical” side of the system, the behaviorally relevant steps are choice set generation and choice (Bowman and Ben-Akiva, 1998).

Choice set generation: Often, the new routes are best paths based on the last iteration (“best reply” or “best response” choice set generation). The routes are generated within the iterations because an a priori enumeration of all possible routes is computationally unfeasible.

Choice: Usually, demand is shifted among the routes to improve consistency with the route choice model, assuming—in the simplest case—constant network delays: In a UE, the flow on the current best routes is increased at the cost of the other route flows (“best reply” or “best response” choice), whereas for an SUE, flows are shifted towards the desired route choice distribution (often a version of multinomial logit, e.g., Dial, 1971; Cascetta et al., 1996; Ben-Akiva and Bierlaire, 1999). For stability reasons, this shift is typically realized in a gradual way that dampens the iteration dynamics. See below for more discussion on convergence issues.

The **iterations** are repeated until some stopping criterion is fulfilled, indicating that a fixed point is attained. In the best reply situation, the fixed point implies that no shift between routes takes place, i.e., what comes out as the best reply to the previous iteration is either the same, or at least of the same performance, as what was used in the previous iteration. Since, in this situation, no O-D pair can unilaterally improve by switching routes, the system is at a Nash equilibrium (e.g., Hofbauer and Sigmund, 1998). In the SUE situation, the fixed point means that a route flow pattern $\{r_k^{od}\}$ is found that leads to exactly those network conditions the travelers (the O-D flows) perceived when choosing their routes, giving no incentive to re-route.

Destination choice and elasticity in the demand are behavioral dimensions beyond route choice that can be captured by a static model. However, no technical generality is lost when discussing only route choice; both additional choice dimensions can be rephrased as generalized routing problems on an extended network (“supernetwork”; see, e.g., Sheffi, 1985; Nagurney and Dong, 2002).

47.2.2 Dynamic Traffic Assignment

The process above also works for *dynamic* traffic assignment (DTA; see Peeta and Ziliaskopoulos, 2001), where both demand and network conditions are time-dependent and the time-dependent travel times in the network define a physically meaningful progression of a demand unit through the network.

The algorithm structure does not change. The individual steps now look as follows:

Algorithm 47.2 Macroscopic and dynamic route assignment

1. **Initial conditions:** Compute some initial routing (e.g., best path on empty network for every O-D pair and departure time).
 2. **Iterations:** Repeat the following many times.
 - (a) **Network loading:** Load all demand items on the network according to their departure times, let them follow their routes and obtain network delays (congestion).
 - (b) **Choice set generation:** Compute new routes based on the network delays.
 - (c) **Choice:** Distribute the demand between the routes based on the network delays.
-

Once more, *if* the new routes are best replies (i.e., best paths based on the last iteration), *if* demand is shifted towards these new routes and *if* these iterations reach a fixed point, then this is then a dynamic UE since best reply dynamics mean that no traveler (no O-D flow) can unilaterally deviate to a better route. The SUE interpretation carries over in a similar way.

Destination choice and elasticity in demand apply naturally to the dynamic case as well. Beyond this, the dynamic setting also enables the modeling of departure time choice. Again, the sole consideration of route choice does, at least technically, not constitute a limitation because departure

time choice can be translated into route choice in a time-expanded version of the original network (van der Zijpp and Lindveld, 2001).

47.2.3 Individual Travelers

In both the static and dynamic case, it is possible to re-interpret the algorithm in terms of individual travelers. In the static case, for every O-D pair, one needs to assume a steady (= constant) flow of travelers entering the network at the origin at a constant rate, corresponding to that O-D flow. A solution to the static assignment problem corresponds to the distribution of different travelers onto possibly different paths.

In the dynamic case, one needs to generate the appropriate number of travelers for every O-D pair and every time slot and distribute them across the time slot. From then on, the triple (origin, destination, departure time) is fixed for every simulated traveler; its goal is to find an appropriate path. Arguably, this re-interpretation is behaviorally more plausible in the dynamic case.

In a trip-based context, there are two major motivations to go from continuous flows to individual travelers:

- Traffic flow dynamics in complex network infrastructures are difficult to model as continuous flows (e.g., Flötteröd and Rohde, 2011), but are relatively straightforward to simulate at the individual vehicle level (TSS Transport Simulation Systems, accessed 2015; Quadstone Paramics Ltd., accessed 2015; Caliper, accessed 2015; PTV AG, accessed 2015; DynusT, accessed August 2014; Zhou and Taylor, 2014). Disaggregating an O-D matrix into individual trip-makers allows the assignment of one vehicle to every trip-maker in the microscopic traffic flow simulation.
- It is computationally inefficient to capture demand heterogeneity through a large number of commodity flows, but the sampling of trip-makers with different characteristics is fairly straightforward. For example, every vehicle can be given an individual route to its individual destination.

For a finite population of heterogeneous travelers, each traveler constitutes an integer commodity and the **choice** step thus must be changed from “gradually shift the route flows towards something consistent with the behavioral model” into “for a *fraction* of travelers, assign a *single* behaviorally plausible route to each of these travelers”. The gradual shift that helps stabilize iterations in the continuous assignment carries over here to an equally stabilizing “inert shift”; not all travelers change their routes at once. This is a consistent reformulation; if one reduces the traveler size from one to $\varepsilon \rightarrow 0$ and increases the number of travelers by a factor of $1/\varepsilon$, a 10 % chance of changing routes in the disaggregate case carries over to shifting 10 % of all flows to new routes in the aggregate case (“**continuous limit**”).

Apart from this, the iterations do not look much different from what was shown before:

Algorithm 47.3 Microscopic and dynamic route assignment

1. **Initial conditions:** Compute some initial routing (e.g., best path on empty network for every traveler).
 2. **Iterations:** Repeat the following many times.
 - (a) **Network loading:** Load all travelers on the network according to their departure times, let them follow their routes and obtain network delays (congestion).
 - (b) **Choice set generation:** Compute new routes based on the network delays.
 - (c) **Choice:** Assign every traveler to a route (which can be the previously chosen one) based on network delays.
-

UE and SUE notions carry over to the disaggregate case if the notion of an O-D pair (or a commodity) is replaced by an individual **particle** (= microscopic traveler).

A **particle UE** may be defined as a system state where no particle can unilaterally improve itself. This definition is consistent with definitions in game theory, which normally start from the discrete problem. It should be noted, however, that this makes the problem combinatorial, which means that even a problem that had a unique solution in its continuous version may have a large number of solutions in its discrete version. Further, the complexity of finding a solution increases similarly to the situation where linear programming jumps to being NP-hard when the variables are required to be integers.¹ The particle UE is hence deliberately not searching for an integer approximation of the continuous solution.

Situations may occur where mixed strategy equilibria exist; these are equilibria, where participants draw between different fixed strategies randomly. This implies that the opponents need to interpret the outcome of the game probabilistically: Even if they themselves play fixed strategies, they need to maximize some expectation value.

For a **particle SUE**, the continuous limit assumption of the macroscopic model is discarded in that the choice fractions $P_k^{od}(c(x(\{r_k^{od}\})))$ in (47.3) are now interpreted as individual-level choice probabilities $P_k^n(c(x(\{r_k^n\})))$ where r_k^n is a binary variable that indicates if traveler n takes route k or not. This implies that the individual-level route flows r_k^n are now random 0/1 variables; consequently, the cost structure—based on individual choices made—becomes probabilistic as well (Balijepalli et al., 2007; Cascetta and Cantarella, 1991; Cascetta, 1989).

A particle SUE is defined as a system state where travelers draw routes from a stationary choice distribution and where the resulting distribution of traffic conditions re-generates that choice distribution.

An operational **particle SUE** specification results if one assumes that travelers filter out the random fluctuations from what they observe and base their decisions only on average route costs:

$$P_n(k) = P_n(k | E\{c(x(\{r_k^n\}))\}) \quad (47.4)$$

where $P_n(k)$ now is the probability that trip-maker n selects route k and $E\{\cdot\}$ denotes the expectation. This approach incorporates some generality; it can be shown that choice distributions based on expected network conditions coincide, up to first order, with the stationary choice distributions based on fluctuating network conditions (Flötteröd et al., 2011).

The resulting route flows r_k^n represent not only mean network conditions but also their variability, due to the individual-level route sampling. Alternatively, one could use the particles merely as a discretization scheme of continuous O-D flows and distribute them as closely as possible to the macroscopic average flow rates (e.g., Zhang et al., 2008). The latter approach, however, does not lend itself to the subsequently developed behavioral model type.

No new behavioral dimensions are added when going from commodity flows to particles. However, the microscopic approach allows simulation of greater behavioral variability within the given choice dimensions because it circumvents the computational difficulties of tracking a large number of commodity flows. This will be discussed in more detail in Section 47.2.5.

47.2.4 Stochastic Network Loading

The network loading can be deterministic or stochastic. With deterministic network loading, given time-dependent route inflows, one obtains one corresponding vector of network costs. With stochastic network loading, given the same input, one obtains a *distribution* of vectors of network costs.

¹ See http://en.wikipedia.org/wiki/Linear_programming_relaxation.

The macroscopic SUE approach of Section 47.2.1 assumes a distribution of choices but converts choice probabilities into choice fractions before starting the network loading. That is, one effectively performs $\text{NetworkLoading}(E\{\text{Choices}\})$. It is, however, not clear that this is the same as $E\{\text{NetworkLoading}(\text{Choices})\}$; in fact, with a non-linear network loading, even when it is deterministic, the two are different (Cascetta, 1989). Any Monte Carlo simulation of the particle SUE makes this problem explicit: If, at the choice level, one generates draws from the choice distribution, it makes sense to *first* perform the network loading and *then* do the averaging, rather than the other way around. This is especially true if day-to-day replanning is modeled, in which case draws from the choice distribution have a behavioral interpretation as the actual choices of the trip makers on a given day (but see also Section 97.3.5).

This, however, makes the output from the network loading effectively stochastic since the input to the network loading is stochastic. In consequence, any behavioral model that uses the traffic conditions as input needs to deal with the issue that these inputs are stochastic. *Thus, using a stochastic instead of a deterministic network loading makes little difference.* Making the network loading stochastic simplifies the implementation of certain network loading models. In particular, randomness is a method to resolve fractional behavior in a model with discrete particles.

With stochastic network loading, additional aspects of the iterative dynamics need to be defined. For example, a “best reply” could be against the last stochastic realization or against some average.

47.2.5 Extending the Route Assignment Loop to Other Choice Dimensions

Given the above behavioral interpretation, it is now straightforward to extend the assignment loop to other choice dimensions. For example, the “best reply” can include optimal departure time choice (e.g., de Palma and Marchal, 2002; Ettema et al., 2003) or optimal mode choice. This becomes easiest to interpret (and, in our view, most powerful in practice) if one moves from the concept of “trips” to daily plans. MATSim plans maintain the structure of DTA in terms of the triple (origin, departure time, destination); see Section 2.2.2.3 for an example. However, different from DTA, all activities are chained together.

This widens the behavioral modeling scope dramatically; all choice dimensions of an all-day travel plan can now be jointly equilibrated. This increases the degrees of freedom that need to be modeled but also carries a set of natural constraints along, which again reduce the solution space. Most notably, the destination of one trip must be the origin of the same agent’s (synthetic traveler’s) subsequent trip and an agent must arrive before it departs. Also, constraints such as Hägerstrand’s space-time prisms (Hägerstrand, 1970) are automatically enforced when the agents need to return to their starting locations.

There is so significant conceptual difference between the network loading of a route-based and a plan-based model.

The notion of a particle (S)UE can now be naturally extended to agents that execute complete plans.

An **agent-based UE** implies individual travelers (Section 47.2.3), additional choice dimensions (Section 47.2.5) and possible stochastic network loading (Section 47.2.4). Corresponding to the particle UE, it is defined as a system state where no agent can unilaterally improve its plan.

An **agent-based SUE** implies individual travelers (Section 47.2.3), additional choice dimensions (Section 47.2.5) and, normally, stochastic network loading (Section 47.2.4). Corresponding to the particle SUE, it is defined as a system state where agents draw from a stationary choice distribution and where the resulting distribution of traffic conditions re-generates that choice distribution.

If the iterations aim at an agent-based UE, then choice set generation and choice should implement a “best reply” logic; some ‘optimal’ plans are calculated and assigned to the agents. This is anything but an easy task.

The disaggregate counterpiece of an SUE implies that every agent considers a whole choice set of (possibly suboptimal) plans and selects one of these plans probabilistically, which can lead to huge data structures.

Summarizing, we have now arrived at a fully disaggregate dynamic DTA specification that accounts for arbitrary behavioral dimensions.

47.3 Agent-Based Simulation

The conceptual validity of the agent-based traffic assignment model is fairly intuitive. However, since it comes with a substantial computational burden of solving the model, it presents entirely new challenges on the simulation side.

On the demand side, there is, in particular, the combinatorial number of choice alternatives that must be considered. For example, random utility models rely on an a-priori enumeration of a choice set that represents options each traveler considers when making a choice (Ben-Akiva and Lerman, 1985). This choice set is huge in an agent-based simulation (Bowman and Ben-Akiva, 1998). While there are sampling-based approaches to the modeling of large choice sets that aim at reducing this computational burden, they have not yet been carried over to the modeling of all-day-plan choices (Ben-Akiva and Lerman, 1985; Frejinger et al., 2009b; Flötteröd and Bierlaire, 2013). See also Chapter 49.

As long as household interactions are not included, the demand modeling problem can be decomposed by agent once the network conditions are given—a great computational advantage. The supply model, on the other hand, deals with congestion, which is, by definition, a result of all travelers' physical interactions. Modeling large urban areas requires dealing with millions of travelers, and an operational supply simulation must be able to load all of these travelers on the network in reasonable computation time.

The following sections describe solutions for these problems. Concrete examples of much of this material are implemented within MATSim.

47.3.1 Agent-Based UE; One Plan per Traveler

The simulation of an agent-based UE is possible through the following implementation of the behavioral elements.

Choice set generation: For every agent, generate what would have been best in the previous iteration. This does not concern just the route but all considered choice dimensions, e.g., departure times and/or mode choice.

Choice: Switch to the new plan with a certain probability.

The choice set generation implements a “best reply” dynamic. This now requires identification of an optimal all-day plan for given network conditions. While the calculation of time-dependent shortest paths for UE route assignment is computationally manageable, the identification of optimal plans is far more difficult (Recker, 2001). This is an important technical motivation to switch to an agent-based SUE, where optimality is not required (see below).

Even in the manageable cases of, e.g., shortest paths, any best reply computation is an approximation. Time-dependent routing algorithms require knowledge of every link's travel time as a function of the link entrance time. In computational practice, this information exists only in an average, interpolated way. Thus, such computations become more robust if plan performance is directly taken from the network loading instead of relying on the best reply computation prediction; an agent sticks with a new plan only if it *performs* better than the previous plan (Raney and Nagel, 2004). However, to keep run times manageable in computational practice, multiple agents

must make such trial-and-error moves simultaneously. This is, therefore, not an exact best reply algorithm.

For the choice, a useful approach is to make the switching probability from current to best reply solution proportional to the expected improvement, i.e.,

$$P(\text{old} \rightarrow \text{best}) = \min[1, \mu \cdot (S_{\text{best}} - S_{\text{old}})]$$

where S_{best} and S_{old} are the (expected) scores of the best reply and the old plan, respectively, and the min takes care of the fact that a probability should not be larger than one. Truncation at zero is not necessary because the term $S_{\text{best}} - S_{\text{old}}$ cannot become negative. Chapter 3 gives an example of what a scoring function for all-day plans could look like. Note how the decreasing switching *fraction* of the continuous case is replaced by a decreasing switching *probability* (leading to a switching *rate*).

Clearly, any fixed point of such iterations is a UE since no switching takes place at the fixed point, meaning that the best reply plan has the same score as the already existing plan. Stability of the fixed point depends on the switching rate slope at the fixed point, in the above formulation on the μ : All else equal, making μ smaller makes the fixed point more stable but slows down convergence. These observations hold not only in transportation (e.g., Watling and Hazelton, 2003) but quite generally in the area of “evolutionary games and dynamical systems” (Hofbauer and Sigmund, 1998). In addition, in the context of traffic assignment, the existence of physical queues allowing for spillback across many links has apparently been shown to be an inevitable source of multiple Nash equilibria (Daganzo, 1998).

Alternatively, some MSA-like scheme may be used (Liu et al., 2007). One disadvantage with MSA is that the switching rate does not depend on the magnitude of the expected improvement, which could mean slow(er) convergence. An advantage of MSA is that one does not need to find out a good value for the proportionality factor (μ in the above example).

Yet another approach would be to use a “gap” function measuring the distance of the current assignment from an equilibrium and to infer the switching rate from the requirement that this function must be minimized (Lu et al., 2009; Zhang et al., 2008). However, we are not aware of any operational gap function that applies to all-day plans.

The biggest criticism of agent-based UE is its lack of behavioral realism. In a UE, every agent is assumed to react with a best response according to a model of its objectives, which implies that real travelers are able to compute best responses despite their combinatorial nature and high dimension (Bowman and Ben-Akiva, 1998). Furthermore, as in a pure route assignment, it is reasonable to assume that (i) the behavioral objective is imperfectly modeled and that (ii) explorative travel behavior leads to—more or less—random variations in what real travelers do. While (ii) explicitly introduces stochasticity, (i) calls for it as a representation of imprecision in the behavioral model.

These considerations do not only lead naturally to the agent-based SUE; they also stimulate an additional behavioral component capturing real travelers’ explorative learning. Similar to the symmetry between day-to-day replanning and the traffic assignment problem’s iterative solution, an explorative learning algorithm can be interpreted either as a model of real learning or as a computational method to solve a stochastic assignment problem. The following section presents a possible implementation of such an algorithm.

47.3.2 Agent-Based SUE; Multiple Plans per Traveler

This section discusses MATSim’s co-evolutionary algorithm for simulating plan choices. Chapters 49 and 51 provide an alternative perspective on MATSim’s plan choice mechanisms in terms of mainstream discrete choice theory (Ben-Akiva and Lerman, 1985).

It is possible to approach every agent's daily planning problem as a population-based search algorithm. Such a search algorithm maintains a *collection* (= population) of possible solutions to a problem instance and obtains better solutions via that collection's evolution. This is a typical machine-learning (e.g., Russel and Norvig, 2010) approach; the best-known population-based search algorithms (also called **evolutionary algorithms**) are **genetic algorithms** (e.g., Goldberg, 1989).

It is important to note that “population” here refers to the collection of solutions for a single individual, as opposed to the population of travelers. Every individual uses a population-based algorithm to “co-evolve” in the population of all travelers (also see Balmer, 2007).

A **population-based search algorithm** typically works as follows:

Algorithm 47.4 Population-based search

1. **Initiation:** Generate a collection of candidate solutions for a problem instance.
 2. **Iterations:** Repeat the following many times.
 - (a) **Scoring:** Evaluate every candidate solution's “score” or “fitness”.
 - (b) **Selection:** Decrease the occurrence of “bad” solutions. There are many ways how this can be done.
 - (c) **Construction of new solutions:** Construct new solutions and add them to the candidate solutions collection.
-

For the construction of new solutions, two operators are often used in genetic algorithms: **Mutation**—which takes a candidate solution and performs small modifications to it; and **crossover**—which takes *two* candidate solutions and constructs a new one from those. Since mutation takes one existing solution and crossover takes two, it makes sense to also move in the opposite direction and define an operator that takes zero solutions as input, i.e., generates **solutions from scratch**—a “best-reply to last iteration” would, for example, be such an operator.

Like what has been said before, we typically have a situation where multiple travelers evolve simultaneously: a *population of persons* where every person has a *population of plans*. The result is a co-evolutionary dynamic, where each person evolves according to a population-based **co-evolutionary algorithm**. The overall approach reads as follows (see, e.g., Hrabner et al., 1994; Arthur, 1994, for similar approaches):

Algorithm 47.5 Co-evolutionary, population-based search

1. **Initiation:** Generate at least one plan for every agent.
 2. **Iterations:** Repeat the following many times.
 - (a) **Selection/Choice:** Select one of the plans for every agent.
 - (b) **Scoring:** Obtain a score for every agent's selected plan by executing all selected plans simultaneously in a simulation and attaching some performance measure to each executed plan. Clearly, what was previously the network loading has now evolved into a full-fledged agent-based simulation of daily activities. See Section 47.3.2.4 for more detail on scoring.
 - (c) **Generation of new plans (innovation)/Choice set generation:** For some of the agents, generate new plans; for example, as “best replies” or as mutations of existing plans (e.g., small departure time changes).
-

Note that this approach is really quite congruent with the SUE approach: Each person has a plan collection, which may be interpreted as the choice set. As in SUE, the choice set can be generated while the iterations run or before the iterations start. Each person selects between the plans, where one can attach to every plan a score-based probability to be selected, which becomes in the end similar to Equation (47.3). Clearly, a relevant related research topic is to specify an evolutionary dynamic that can be shown to converge to choice sets that are generated consistently with discrete choice theory requirements; see Chapter 49 and Section 97.3.

The following subsections give examples for the different elements of this approach.

47.3.2.1 Selection (Choice)

A possible choice algorithm is the following: For persons with unscored plans, select an unscored plan. For all other persons, select between existing plans with some SUE model, e.g., a logit model, i.e.,

$$P(i) = \frac{e^{\mu S_i}}{\sum_j e^{\mu S_j}} \quad (47.5)$$

where S_i is the score of plan i and μ models the travelers' ability to distinguish between plans with different scores. This is implemented in MATSim by `SelectExpBeta`.

In practice, we have found that it is much better to not use Equation (47.5) directly but instead use a switching process that *converges* towards Equation (47.5). This can, for example, be achieved by using a switching probability from i to j of the form

$$T(i \rightarrow j) = \gamma e^{\beta(S_j - S_i)/2} \quad (47.6)$$

where i is the previous plan, j is a randomly selected plan from the same person and γ is a proportionality constant that needs to be small enough so that the expression is never larger than one (since it denotes a probability). This works because the logit model (47.5) fulfills the detailed balance condition

$$P(i) T(i \rightarrow j) = P(j) T(j \rightarrow i) \quad (47.7)$$

for these $T(i \rightarrow j)$ (e.g., Ross, 2006).² This is implemented in MATSim by `ChangeExpBeta`.

The “switching approach” has additional advantages, including the following:

- Equation (47.6) can be behaviorally interpreted as the probability of switching from plan i to plan j . Plausibly, this probability increases with the magnitude of the improvement. For certain applications, one might want a more involved approach, e.g., an *expected* score of j , which then initiates the switch.
- One could replace Equation (47.6) by a threshold-based dynamics, i.e., a switch to a better solution will only take place if the improvement is above a certain threshold. One loses some of the mathematical interpretation, but it may be more consistent with discussion of project appraisal, where small improvements may not lead to a change in behavior.

Although not performed systematically in past work, it is possible to include formulations such as path-size logit (Ben-Akiva and Bierlaire, 1999) in the choice model.

² Assume that, after a number of iterations, there is no more innovation, i.e., the choice set for every agent is fixed and that the scores are updated by MSA. On convergence of the iterations, all agents draw their plans from a fixed choice set based on constant score expectations, cf. (47.4). This means that all agents make their choices independently (and that all interactions are captured in the scores). The switching logic (47.6) then defines an ergodic Markovian process, which converges to the unique steady state probabilities (47.5).

47.3.2.2 Score Convergence

The assumption that the scores eventually converge to some constant value intuitively means that the scores cannot display spontaneous reactive behavior to a certain iteration. For example, a particular iteration might display a “network breakdown” (Rieser and Nagel, 2008). Converged scores would not trigger a next-day reaction to that breakdown. In practice, this can be achieved by averaging the scores over many iterations, which is somewhat similar to fictitious play (Monderer and Shapley, 1996; Garcia et al., 2000). Once more, MSA is an option (Section 3.3.4), with the same advantages and disadvantages discussed before (Section 47.3.1). An alternative is to use a small **learning rate** α (Section 3.3.3) in

$$S_i^{\text{new}} = (1 - \alpha)S_i^{\text{old}} + \alpha \tilde{S}_i \quad (47.8)$$

where S_i^{new} and S_i^{old} are the agent’s memorized scores for option i , and \tilde{S}_i is the most recent actual performance with that option; also see Chapter 49. The issue, in the end, is the same as the stable-vs-unstable fixed points (cf. Section 47.3.1): If the system is well-behaved (corresponding to a stable fixed point), it will converge benignly to constant scores and thus to the detailed balance solution. If the system is not well-behaved, one can still force it to such a solution with MSA, but the meaning is less clear (also see Sections 3.3.4 and 47.3.2.2).

As stated before, stochastic network loading makes no additional conceptual difference since there is already stochasticity caused by choice behavior.

47.3.2.3 Innovation (Choice Set Generation)

So far, this leaves open the question on choice set *generation*, i.e., the part that generates new plans or modifies existing ones.

One computationally simple technique not requiring a choice set enumeration is to simulate randomly disturbed link costs and run best response based on these costs. This, however, can yield unrealistic results if one does not get the correlation structure of the noise right.

An alternative is to calculate separate best responses after every network loading. Since the process is stochastic, this will generate different solutions from iteration to iteration. An advantage is that the correlations will be generated by the simulation—and are, presumably, realistic. Chapter 49 relates this to random utility modeling; see also Chapter 97.

Beyond that, there are many different algorithms that could be used here. Besides the previously-mentioned “mutation” (Balmer et al., 2005b) or “crossover” (Charypar and Nagel, 2005; Meister et al., 2006), there are also many possibilities for constructive algorithms, such as “agent-based” construction (Zhu et al., 2008). One attractive option, clearly, is to use a regular activity-based demand generation code (e.g., Bowman et al., 1998; Miller and Roorda, 2003), although we have found that this may not be as simple as it seems (Rieser et al., 2007b); in practice, activity-based models are often constructed with O-D matrices in mind. A successful integration is described by Ziemke et al. (2015).

47.3.2.4 Adjusting the “Improvement Function” from Shortest Time to Generalized Utility Functions

This chapter takes an inductive approach and argues that one can make the network assignment loop more general by including additional choice dimensions beyond routing. Clearly, for this to work, the scoring needs to take the effects of these additional choice dimensions into account (also see Balmer, 2007). Given evolutionary game theory, it is quite obvious how to do that: One has to extend the cost function used for routing to a general scoring function for complete daily plans.

That is, the performance of a daily plan needs to be scored. An established method to estimate scoring functions for different alternatives is random utility theory (e.g. Ben-Akiva and Lerman, 1985), which is why in the following “scoring” will be replaced by “utility”. For a utility function for daily plans, the following arguments may serve as starting points:

- A heuristic approach, consistent with wide-spread assumptions about travel behavior, is to give positive rewards to performing an activity and negative rewards to traveling.
- For the activities, one should select functions where the marginal reward of doing an activity decreases over time.
- Without additional effects, such as opening times or time-varying congestion, the marginal utilities of all performed activities should be the same.

MATSim has, in the past years, gained some experience with the approach described in Chapter 3 and with more theory in Chapter 51; this then closes the loop.

47.4 Conclusion

Starting from regular route assignment, this chapter explains how one can extend the iterative solution procedure of static or dynamic traffic assignment to include additional behavioral dimensions such as time adaptation, mode choice or secondary activity location choice. This is somewhat similar to the so-called supernetworks approach but argues from the viewpoint of the iterative solution procedure rather than the problem definition.

To address the combinatorial explosion of commodities caused by the expansion of the choice dimensions, a move to individual particles is suggested. This allows an interpretation of the solution procedure as behavioral day-to-day learning but maintains a connection to the SUE definition by interpreting synthetic travelers’ behavior as random draws from individual choice sets.

Most of this chapter discusses simulation/computer implementation issues. From the definition given above, progress can be made by using methods from machine learning and co-evolutionary search algorithms. The SUE problem of random selection between different alternatives can be cast as a so-called population-based optimization algorithm, where each synthetic traveler randomly selects between the different members of the population of possible solutions. At the same time, the *population of the travelers* co-evolves towards a stationary distribution of choices.

Overall, this chapter has worked out the structural similarity between the “classical” DTA problem and the more recent agent-based assignment problem.³ The presentation has focused on the algorithmic issue of how to find solutions to these problems. This is complemented by the subsequent Chapters 48 to 50, which mostly discuss modeling (descriptive) aspects of MATSim.

³ It is, in fact, possible to run MATSim in DTA mode, by converting each trip into a dummy person, with dummy activities at the beginning and end of the trip. The class `RunExample5Trips` (see <http://matsim.org/javadoc> → main distribution) runs an example; the class itself points to a configuration file, which in turn points to `examples/equil/plans100trips.xml`. A dummy person that denotes a trip from link 1 to link 20, departing at 6 am, is coded as

```
<person id="1">
  <plan>
    <act type="dummy" link="1" end_time="06:00" />
    <leg mode="car" />
    <act type="dummy" link="20" dur="00:10" />
  </plan>
</person>
```