Introducing joint trips in a multi-agent transport simulation: from agents to clique replanning

Diploma Project, Urban Systems Engineering

Projet de Fin d'Études, Génie des Systèmes Urbains

Thibaut Dubernet

UTC supervisor:

Fabrice Locment

IVT Supervisors:

Francesco Ciari

Kay W. Axhausen

July 2011

Institut für Verkehrsplanung und Transportsysteme
Institute for Transport Planning and Systems

utc
Université de Technologie
Compiègne

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Abstract

MATSim is a multi-agent transport simulation software, based on an iterative relaxation procedure. In this framework, each agent is assigned a daily plan consisting of a sequence of trips and activities, to which a score is associated. Daily plans are optimised knowing the state of the traffic flows. The system iterates between traffic flow simulation and plan modification, until an equilibrium is reached.

In this report, the ability for the agent to travel together with other agents is introduced. This is done by passing from optimisation of the scores of individual plans to the optimisation of scores associated to joint plans. To perform such an optimisation, a genetic algorithm is used, which is presented.

This algorithm is able to improve the scores of a joint plan by identifying pertinent joint trips. However, clues show remaining convergence difficulties, which are analysed.

Résumé

MATSim est un logiciel multi-agent de simulation du traffic. Le principe de base est le suivant: chaque agent possède un plan, correspondant à une séquence de trajets et d'activités. Ces plans sont optimisés individuellement, sachant les conditions de traffic, au cours d'un processus itératif: en alternant simulation du traffic et modification des plans individuels, le système atteint un équilibre, correspondant à l'état recherché.

Dans ce rapport, les agents se voient donner la possibilité de partager leurs véhicules avec d'autres agents. Ceci se fait en passant de l'optimisation de plans individuels à l'optimisation de plans joints, se voyant affecter un score global. Pour mener à bien cette optimisation, un algorithme génétique a été utilisé.

Cet algorithme, bien que capable d'améliorer les scores d'un plan joint en séléctionnant des trajets joints pertinents, présente encore quelques problèmes de convergence. Une analyse de ce fait est proposée.

Contents

List of Figures

List of Tables

Introduction

Traffic simulation models are used to predict traffic flows on a network, aiming at supporting analysis and decision taking.

Since the first models proposed in the middle of the twentieth century, the increase in computational power and the continuous improvement of traffic models led to always preciser and finer predictions. A successful framework for simulating traffic is multi-agent activity based transport simulation, where agents, representing individuals, travel through a simulated network from one activity to the other.

With such models, the simulation is based on behavioural models, and thus virtually allows to simulate any behaviour which impact on traffic is assumed, or known, to be important. The fact that several persons may coordinate themselves to travel together is such a behaviour. Its simulation can be used to predict the impact of car-pooling on traffic flows, or the impact of car-pooling incentives on the share of this mode. Another major application is to better simulate the behaviour at the household level, where the share of such trips is known to be significant.

In this report, we present an algorithm aimed at simulating such trips in MATSim, a multi-agent traffic simulation software. This algorithm is designed to generate plans for group of agents, with joint trips.

In Chapter 1, we review the literature related with activity based transport simulation and generation of schedules for groups or persons. In Chapter 2, the proposed algorithm is presented. Chapter 3 presents simulation experiments aiming at testing the behaviour of the algorithm.

1 Activity-based traffic simulation and joint decision modeling

1.1 Activity-based travel simulation: basic concepts and presentation of MATSim

1.1.1 The activity-based approach

Simulation of travel behaviour is a widely used tool, which can be used for predicting the effects of some change infrastructure, reconstruct missing data about the current state, or for policy evaluation.

Generally, simulation models are classified as macroscopic, mesoscopic or microscopic, depending on the level of aggregation used.

Naturally, each type of model has its strength and weaknesses. While macroscopic models, which only work at the aggregated level, are computationally efficient and only require aggregated data as input, they have difficulties to represent time-varying aspects of traffic. On the other hand, microscopic models, by simulating agents individually, can predict traffic dynamics much more easily, but at a much higher computational cost and with finer data as input.

However, increase in computational power in the last decades has made this kind of models more and more appealing.

A successful framework while simulating individuals at a disaggregated level is to use the so-called "activity-based " approach, proposed during the early eighties (Jones *et al.* (1983); Recker (1986)). In this approach, the fact that travel is always oriented toward a goal is taken into account explicitly: agents are assigned plans, consisting of located activities, and travel between those activities in a simulated network. A fundamental difference with trip-based approaches is the explicit modeling of travel as a need *derived* from the need or willingness to perform activities (McNally (2000)).

The way the plans are computed depends on the model: in the following, we focus on the MATSim software.

1.1.1.1 Equilibrium based models: the MATSim process

MATSim is an open-source software, which mainly aims at producing pertinent plans for agents, plans which are used in a traffic simulation (Balmer *et al.* (2008); Rieser *et al.* (2007)). By plan is meant a

succession of activities and trips between those activities, beginning and ending at the home location of the agent. Those plans are understood as being daily plans.

To produce such plans, MATSim uses the assumption that the state of the real system corresponds to a Nash equilibrium. This fundamental concept of game theory is defined thereafter.

Definition 1 (game in normal form)**.** *A **game in normal form** is a tuple* $\left\{N, (S_i)_{i\in N}, (u_i)_{i\in N}\right\}$*, where* $N = \{1, 2, \ldots, n\}$ *is the set of players,* S_i *is the action space of player* i *and* $u_i : S \to \mathbb{R}$ *is the utility function for player* i*, where* $S = \times_{j\in N} S_j$*. The* i^{th} *element* s_i *of* $\mathbf{s} \in S$ *is named the **strategy** of player* i*.*

Thus, a game associates individual utilities to the action of the whole set of players. In the case of traffic, this dependence on other players' choices is due to the effect of congestion.

Definition 2 (Nash equilibrium)**.** *Let* $(s'_i \mid \mathbf{s})$ *be the group action* s *where the individual action of player* i *is changed to* s'_i*.*

A point $\mathbf{s} \in S$ *is a **Nash equilibrium** if and only if:*

$$\forall i \in N,\ \forall \mathbf{s}' \in S,\ u_i(\mathbf{s}) \geq u_i(s'_i \mid \mathbf{s})$$

That is, no player can improve its utility by unilaterally modifying its strategy.

In a simple routing game (without time effects: cost of a link is a function of the number of players choosing it), the cost of all flows at equilibrium in an network without capacity restriction and with differentiable and convex cost function is known to be unique (Nisan *et al.* (2007)).

Koch and Skutella define the Nash equilibrium for time varying traffic flows as a flow which state is a Nash equilibrium at any point in time, and prove the existence of such an equilibrium in the case of a flow between a unique origin/destination pair (Koch and Skutella (2009)). However, they do not consider departure time as part of the strategy.

However, to our knowledge, there exists no result general enough to be applied to a complex plan-based microsimulation model. If aggregate properties of traffic flows (as the number of vehicles passing through a link in a given period) are likely to be equivalent in different Nash equilibria, the absence of uniqueness result implies to be careful while analysing disaggregate output from a simulation. That is, agents having different strategies in two different equilibria, trying to analyse individually each strategy may result in non-sense.

A classical way to compute a Nash equilibrium for a given game is to use a relaxation algorithm: strategies of players are iteratively "improved", given the previous strategies of other players, until no further improvements are possible. To ensure convergence, only some player's strategies are improved, or only suboptimal improvement are made for all players (Berridge and Krawczyk (1998)).

Within MATSim, the strategies of the individuals corresponds to their plans, and the utility function is calculated based on the results of a traffic flow simulation. The relaxation takes the following form:

starting with initial plans, agents are moved through a simulated network, giving estimates of the cost of travel. Then, plans of a given fraction of the agents are mutated, randomly or to optimality given the previous state. Non mutated agents choose one of their previous plans based on the past scores, and the simulation is run again. This process is executed until a stopping criterion is met (currently, a fixed number of iterations fixed *a priori* is used (Meister *et al.* (2010))).

This process allows to take into account nicely the complex relationship between traffic flows and the utility of a plan.

Algorithm 1 the MATSim process

```
load initial plans;
repeat
    run traffic simulation;
    compute plan scores;
    modify plans;
until stopping criterion is met
```

It may be important to precise that in the iterative relaxation process, the *same* day is simulated over and over until an equilibrium state is found. That is, even though agents learn in some sense, and even if the actual equilibrium state may result from human learning, one should not try to interpret the evolution process as a simulation of human learning, but rather as a smart way of exploring the space of agents' plans to find an equilibrium.

Of course, the way agents' plans are modified has a significant impact on the results of the optimisation. Currently, replanning can include least-cost re-routing, location choice (Horni *et al.* (2008)), duration and mode optimisation (Meister *et al.* (2005a, 2006)), or activity sequence (Feil *et al.* (2009)).

1.1.1.2 Plan scores in MATSim

As pointed out, MATSim mainly consists in a daily plan optimisation algorithm coupled to a traffic flow simulation. Thus, a performance metric is needed.

In general, MATSim uses the so-called "Charypar-Nagel" scoring function, first introduced to generate daily plans out of the iterative MATSim process (Charypar and Nagel (2005)).

In this formulation, the utility of a plan takes the form of a sum of the activity of performing activities and of the disutility of traveling:

$$F = \sum_{i=1}^{n} U_{act}(\mathrm{type}_i, \mathrm{start}_i, \mathrm{dur}_i) + \sum_{i=2}^{n} U_{trav}(\mathrm{loc}_{i-1}, \mathrm{loc}_i) \tag{1.1}$$

where the utility of an activity is:

$$U_{act,i} = U_{dur,i} + U_{wait,i} + U_{late.ar,i} + U_{early.dp,i} + U_{short.dur,i} \tag{1.2}$$

and:

$$U_{dur}(t_{dur}) = \beta_{dur}\, t^* \ln\left(\frac{t_{dur}}{t_0}\right) \tag{1.3a}$$

$$U_{trav}(t_{trav}) = \beta_{trav}\, t_{trav} \tag{1.3b}$$

$$U_{wait}(t_{wait}) = \beta_{wait}\, t_{wait} \tag{1.3c}$$

$$U_{late.ar}(t_{start}) = \begin{cases} \beta_{late.ar}(t_{start} - t_{latest.ar}) & \text{if} t_{start} > t_{latest.ar} \\ 0 & \text{otherwise} \end{cases} \tag{1.3d}$$

$$U_{early.dep}(t_{end}) = \begin{cases} \beta_{early.dep}(t_{earliest.dep} - t_{end}) & \text{if} t_{end} < t_{earliest.dep} \\ 0 & \text{otherwise} \end{cases} \tag{1.3e}$$

$$U_{short.dur}(t_{end}) = \begin{cases} \beta_{short.dur}(t_{short.dur} - (t_{end} - t_{start}) & \text{if} t_{end} < t_{short.dur} \\ 0 & \text{otherwise} \end{cases} \tag{1.3f}$$

$$\tag{1.3g}$$

where:

t^*	is the typical duration for the activity
t_0	is the minimal duration for the activity
t_{dur}	is the actual utility duration
t_{trav}	is the traveling time
t_{wait}	is the waiting time
t_{start}	is the start time
t_{end}	is the end time

1.1.2 Using optimisation algorithms to replan agents

As pointed out before, the relaxation process consists in iteratively improving the plans of the agents, knowing the previous behaviour of other agents, until a steady state is reached.

At the beginning of MATSim development, the approach to do so was to randomly modify plans. Each agents possesses a memory, in which are stored a fixed number of past plans, allowing to revert changes implying a decrease in utility.

However, another approach has been implemented since then, making use of optimisation algorithms in the mutation step.

This approach was shown to allow MATSim to converge in fewer iterations to an equilibrium state, with a score at least as high as with random mutation. Optimisation algorithms used include least-cost routing, activity duration optimisation with CMA-ES (Charypar *et al.* (2006)) or genetic algorithm (Meister *et al.* (2006)), or activity sequence and other properties with Tabu Search (Feil *et al.* (2009)).

1.2 Joint scheduling models

The random utility theory is a well-known and extensively studied way of predicting individual's behaviour, which is widely used in transportation research (Ben-Akiva and Lerman (1985)). In this general framework, each alternative is associated a numerical utility, composed of a systematic part (its expectation) and a random error term (representing unobserved variables). The probability for an individual to choose one of the alternatives corresponds to the probability for the utility of this alternative to be higher than the utility of all other alternatives.

This framework as been applied to joint decision making, and to joint scheduling in particular: we provide here a review of those studies.

Aside from these random utility models , non-probabilistic utility maximisation techniques have been proposed for creating schedules for activity based transport simulation: we present here some of those attempts for household plans generation.

1.2.1 Random utility based models

The random utility theory as been applied early to joint decision modeling, by considering the choice problem as a group utility maximisation problem.

In the last decades, this framework began to be applied to group (mainly household) schedules generation for activity based transport simulation.

However, the choice set is of high dimension, with both discrete (activity types, joint activity participation, sequence of activities, modes *etc.*) and continuous (activity duration) dimensions. Thus, depending on the authors, different choice dimensions are considered.

Zhang, Timmermans and Borgers develop a model where time for different activity types is allocated to household members, subject to time constraints (including equality of time participation in joint activities) (Zhang *et al.* (2005)). Given individual random utilities for the different activity type, their model gives deterministic time allocation.

Bradley and Vovsha focus on the "daily activity pattern" generation, with household "maintenance" tasks (*e.g.* shopping) allocation and possibility of joint activities (Bradley and Vovsha (2005)). To do so, they

assume a layered choice structure: first, a daily activity pattern is assigned to household members; then, "episodic" joint activities can be generated; finally, maintenance activities are assigned.

Gliebe and Koppelmann (Gliebe and Koppelman (2005)) also base their models on the daily activity pattern concept. In their model, the joint outcome (the succession of individual and joint activities) is first determined, and individuals then choose an individual pattern compatible with the joint outcome. The same authors also derived a constrained time allocation model, which predicts the time passed by two individuals in joint activities (Gliebe and Koppelman (2002)). Rather than postulating a group-level utility function, those models specify a special distribution for the error terms of the individuals. In this setting, the error term of the individuals are correlated so that the probability of choosing a given joint output is the same for all individuals.

Miller, Roorda and Carrasco develop a model of household mode choice (Miller *et al.* (2005)). The main difference with an individual mode choice model is the consideration of household-level vehicle allocation. In their model, individuals first choose modes individually. If a conflict occur, the allocation that maximizes the household level utility is chosen. The members which were not allocated the vehicle will report on their second best choice, and/or examine shared rides options.

1.2.2 Alternative approaches

Aside from the random utility theory based models, some other ways to deal with joint scheduling have been proposed.

Contrary to the previously presented approaches, those alternatives did not lead to estimate a model against data. In fact, those are optimisation algorithms, designed to handle the household scheduling problem by transforming it into a deterministic utility maximisation problem. However, the estimation of the utility function is not part of those approaches.

The first of those approaches was proposed by Recker in the mid nineties (Recker (1995)). By extending increasingly the formulation of the Pick-Up and Delivery Problem With Time Windows, which is a well studied combinatorial optimisation problem, he formulates the problem of optimising the activity sequence of members of an household as a mathematical programming problem, taking into account vehicle constraints, individual and household level activity, possibility of choosing whether to perform or not an activity, with the possibility of shared rides. However, due to the complexity of the problem, the full problem cannot be solved exactly by standard operations research algorithms, and the activity durations are not part of the optimised dimensions.

Golob and McNally propose a structural equation model, which predicts time allocation and trip chaining based on descriptive variables of an household (Golob and McNally (1997)). Golob also used this structural equation model approach to model the dependency of time allocations of the two heads (man and woman) of an household (Golob (2000)).

A recent attempt to generate plans for households uses a genetic algorithm, building on a previous genetic algorithm for individual plan generation (Charypar and Nagel (2005); Meister *et al.* (2005b)). This algorithm optimises sequence, duration and activity choice for an household, rewarding the fact for several members of the household to perform the same activity simultaneously (*i.e.* "joint activities").

Finally, Fang *et al.* optimise location and time of joint activities, constrained to time windows in pre-determined individual plans, using a multi-objective genetic algorithm (Fang *et al.* (2011)).

2 Including joint trips in MATSim

2.1 Problem formulation

We focus here on the following problem: in a traffic microsimulation, how to take into account the ability that individuals have to share a ride in the same vehicle? We present here the formulation, hypotheses and restrictions that we choosed for this problem, to transform it into an optimisation problem.

2.1.1 General thoughts on joint trip participation in microsimulation

In an equilibrium-based traffic microsimulation, one searches the equilibrium state where no agent can improve its utility.

Of course, the improvements that one can reach depend on the degrees of freedom one gives to agents: can an agent change its leisure location? And its work location? Can he choose not to perform an activity?

The common point of all those classical degrees of freedom is that they correspond to *competitive* strategies: each agent tries to improve its own utility, whatever the other agents are doing. This competitive behaviour is at the core of the Nash equilibrium assumption.

Collaborative travel behaviour is related to *collaborative* strategies, in the sense that it is possible only if several agents decide to perform the joint trip together.

This collaborative behaviour introduces a new dimension: an agent may accept to decrease its utility to improve the one of another agent. This is typically the case when a parent drives his child to some activity.

As stated in 1.2, the usual way to take into account this fact is to define a *joint utility*, to maximise for the group. Even though we follow this tradition, the specification (and the existence) of this group utility is problematic. Particularly, it seems likely that no such function can be defined for totally unrelated people: identifying collaborating agents is a challenging task.

Finally, the mere utility maximizing approach may fail to identify realistic ride sharing groups: social relationships, affinities and other phenomena are likely to have a predominant influence of the identity of potential co-travelers.

For the current work, we do not focus on the problem of identifying which agents should travel together, but on the following: *given* that two or more agents may travel together, how to optimise their plans?

2.1.2 From agent to clique replanning

One of the most important modifications to the initial agent-based microsimulation framework introduced by the ability to plan joint trips is that individual plans are made interdependent.

This interdependency between plans takes the form of simultaneity constraints, that is, equality constraints between elements of two or more plans.

This fact has a direct and important consequence on the way replanning is done: several individuals that (may) travel together have to be replanned together, with those constraints taken into account explicitly. Thus, one have to define clear groups of agents where joint trips may take place. Even though most of the previous work has used households as a "natural" group, we preferred in this work stay at the most general concept of clique, understood as the minimal group which plans are interdependent due to ride sharing options.

In fact, while empirical work seems to show that most of the shared rides occur within households, avoiding to include household-specific processes in the formulation allows to use it to study other kinds of shared rides, as car-pooling services for example.

In this work, cliques are considered as statically identified before the iterative replanning process, but dynamic cliques may be possible, for example building on previous work on social networks in MATSim (Hackney (2009)).

By the usage of static cliques, the competitive Nash equilibrium assumption is still the central concept: the players of the game are now cliques, which compete for utility. Agents constituting the same clique may, or may not, choose to collaborate, in order to maximize the clique's utility.

2.1.3 Joint trips within a clique

A joint trip is represented by the succession of (at least) a pick-up and a drop-off activity for each participant, with the following constraints:

- one participant is identified as the driver;
- each planed pick-up is followed by a drop-off in a non-driver plan, and a drop-off is always preceded by a pick-up;
- each pick-up (resp. drop-off) activity of a non-driver participant must be simultaneous to a pick-up (resp. drop-off) in the driver's plan.

2.1.4 Constraints on plan durations

Activity durations in a joint plan are constrained in several ways:

1. activities must have a positive duration;

2. we focus on daily plans: individual plans must have a duration of d, the day duration;
3. if a joint trip is planned, it must occur at the same time of day in the plans of all the participants.

2.1.5 Consistent mode choice

2.1.5.1 The subtour approach

Obviously, trip-based mode choice may lead to inconsistent mode chains, as going to work by bike and coming back by car. A successful approach to avoid such inconsistencies is the subtour approach (Miller *et al.* (2005); Ciari *et al.* (2008)).

The following definitions are based on the model of Miller *et al.* (2005) and the definitions of Axhausen (2008).

Definition 3 (tour)**.** *a **tour** is a sequence of trips that starts and ends at the home location.*

Definition 4 (subtour)**.** *within a **tour**, a sequence of trips that starts and ends at the same location is named a **subtour** . The start/end location is named an **anchor point** .*

Of course, not every mode is constrained to be used on the whole subtour: one may go to work as a car passenger and come back by public transport, for example.

Definition 5 (chain-based mode)**.** *A mode which cannot be changed within a **subtour** , or only for a smaller **subtour** , is named a **chain-based mode** . Basically, those are the modes tied to a personnal vehicle (car, bike,* etc.*), which are moreover only available if the vehicle is available at the departure **anchor point** .*

Even though trip-based mode choice is possible for non-chain-based modes, we only consider here the case of subtour-based mode choice. An exception to this approach is the passenger mode: passenger trips may occur in subtours of any non-chain-based mode.

2.1.5.2 Mode availability

We define mode availability for a given agent recursively, considering joint trip participation and status (driver or passenger) as already determined. This mathematical formulation will allow us to only construct feasible mode choices during the optimisation of a joint plan.

Let $\mathcal{S} = \{1, 2, \ldots, n\}$ be the set of subtours indices, and 0 be the index of a "dummy" tour, containing all the subtours. The subtours set have a tree structure: a given subtour is "father" of all subtours which anchor point belong to it. Thus, it is possible to define a father function $\mathcal{F} : \mathcal{S} \to \mathcal{S} \cup \{0\}$ which associates a subtour to its "father" subtour, that is, the shorter subtour to which belongs this subtour.

We moreover define $\mathcal{C}$ as the set of chain-based modes, $\mathcal{N}$ as the set of non-chain-based modes, and the full mode choice set $\mathcal{M} = \mathcal{C} \cup \mathcal{N}$. We finally denote by $c \in \mathcal{C}$ the "car" mode.

Let define the following binary variables for $i \in \mathcal{S}$:

$$D_i = \begin{cases} 1 & \text{if the agent is driver in subtour } i \\ 0 & \text{otherwise} \end{cases} \tag{2.1a}$$

$$P_i = \begin{cases} 1 & \text{if the agent is passenger in subtour } i \\ 0 & \text{otherwise} \end{cases} \tag{2.1b}$$

$$X_i^{(m)} = \begin{cases} 1 & \text{if mode } m \text{ is choosen for subtour } i \\ 0 & \text{otherwise} \end{cases} \tag{2.1c}$$

We moreover define $X_0^{(m)} = 1 \; \forall m \in \mathcal{C}$ (which makes all chain-based modes available at home).

We still have to define a "driver in this tour" variable:

$$D'_i = \begin{cases} 1 & \text{if } \exists j \in \mathcal{S},\ \mathcal{F}(j) = i,\ D'_j = 1 \\ D_i & \text{otherwise} \end{cases} \tag{2.2}$$

That is, $D'_i = 1$ for all driver subtours and their parent subtours.

From the previous paragraph we note that:

$$D'_i \times P_i = 0 \qquad \forall i \in \mathcal{S} \tag{2.3a}$$

$$\sum_{m \in \mathcal{M}} X_i^{(m)} = 1 \qquad \forall i \in \mathcal{S} \tag{2.3b}$$

that is, one cannot be passenger if he is or will be driver (and thus needs his car) (2.3a), and must choose one and only one mode for each subtour (2.3b).

We can now define $A_i^{(m)}$, the availability of mode m for subtour i:

$$A_i^{(m)} = \begin{cases} X_{\mathcal{F}(i)}^{(m)} \times (1 - P_i) & \text{if } m = c \\ X_{\mathcal{F}(i)}^{(m)} \times (1 - P_i) \times (1 - D'_i) & \text{if } m \in \mathcal{C} \setminus \{c\} \\ (1 - D'_i) & \text{if } m \in \mathcal{N} \end{cases} \tag{2.4}$$

There is always a mode available, as:

1. for a non-driver subtour, all non-chain-based modes are available;

2. for driver subtours, car is always available (and is the only mode available).

With those definitions, a feasible mode choice obeys to:

$$\sum_{m \in \mathcal{M}} \left(X_i^{(m)} A_i^{(m)} \right) = 1 \quad \forall i \in \mathcal{S} \tag{2.5}$$

2.1.6 Scoring a joint plan

Obviously, when trying to optimise the plan of a clique, one needs to identify a function to optimize. The problem at hand is a typical multi-objective optimisation problem, where several functions (here, the individual scores) are to optimize.

Two different approaches can be identified in the literature:

1. Pareto optimisation: an algorithm identifies a *set* of Pareto-optimal solutions. A solution is said Pareto-optimal if there exist no solution that "dominates" it for every fitness function (Horn *et al.* (1994); Van Veldhuizen and Lamont (2000)). One than has to choose among the generated solutions.
2. Transformation into a mono-objective optimisation problem, by constructing a global objective function by combining the individual ones. This have the advantage to produce a unique solution.

As seen in 1.2 , this second solution is often retained in group behaviour modeling, and we follow this tradition.

An often retained form in behaviour modeling is a (multi-)linear combination of the individual utilities. In the case of this test implementation, due to the lack of data to estimate the parameters of such a combination, we choosed to simply express the group utility as the sum of the individual utilities.

Such formulations lead to Pareto-optimal solutions.

We thus formulate the problem as:

$$\underset{P \in \mathcal{P}}{\text{maximize}} \left(\sum_{p \in P} u(p) \right) \tag{2.6}$$

where:

P	is a joint plan, understood as a set of individual plans
p	is an individual plan
$\mathcal{P}$	the set of all plans neighbors to the initial plan. A plan is neighbor of another if it can be obtained by modifying durations, modes and eventually transforming some joint trips into individual trips (thereafter named joint trip participation).

By definition of $\mathcal{P}$, no new joint trip is produced during the optimisation process. Thus, the algorithm must be fed with probable joint trips. This may come from another behavioral model, or from a car-pooling agency algorithm for example, depending on what one wants to simulate.

Pick-up and drop-off activities have the utility of waiting.

2.2 Inclusion in MATSim

2.2.1 General framework

In the basic MATSim approach, agents are egoist: the only kind of interactions they have with each other is by the traffic they generate.

To allow collaboration between agents, new data structures where introduced. The main ones are:

- a **clique** is a group of agents which plans are aggregated into **joint plans** . Each agent belong to one and only one clique. All individuals which plans are interdependent must belong to the same clique, and individuals which plans are not interdependent should belong to different cliques.
- a **joint plan** is the aggregation of plans of individuals of a **clique** . The idea is to clearly identify interdependent individual plans. All strategies are defined at the clique level, so that the MATSim process aims here at optimizing joint plans.

In the current implementation, cliques are defined externally before launching the iterations, and one and only one joint plan is imported per clique. To allow agents to choose between several collaboration schemes, several joint plans could be imported at the beginning.

Of course, joint plans should be simulated in the traffic flow simulation, the resulting joint scores computed and used by the different strategies.

This means that, in the case where several agents travel together, waiting of co-participants should be simulated.

Unfortunately, the current MATSim traffic flow simulation does not allow yet to simulate such behaviors. Instead, unsimulated modes are "teleported" (moved without interaction with the traffic flow simulation) according to the expected travel time. As such a synchronising capability may be available soon, we used a simple heuristic to correct the scores to some extent. This heuristic is presented in Algorithm 2.

Passengers being teleported according to the expected travel time, an obvious drawback is that, for a shared trip, if the car travel time is lower than expected, it will be scored according to the expected time. However, this allows to take into account unexpected traffic jams with a very simple and inexpensive approach.

Algorithm 2 Joint scoring heuristic

```
for planElement ∈ plan do
    if planElement is a shared trip then
        set departure time to latest departure time of the participants;
        set end time to latest end time of the participants;
    end if
end for
```

2.2.2 Replanning module

2.2.2.1 Replanning with Genetic Algorithms

Genetic algorithms are approximate optimisation algorithms inspired by natural evolution, which were introduced more than three decades ago (Holland (1975)). Those algorithms make evolve a "population" of solutions, based on cross-breeding of solutions and random mutation (called genetic operators), and selective competition.

The solutions are represented by strings of values, named "chromosomes ". Whereas the first genetic algorithms only used binary encoding, well suited for combinatorial problems, all kind of values are now used, including floating point numbers .

In the field of multi-agent microsimulation, genetic algorithms have been successfully used, among other things, to produce daily plans for agents and groups of agents. They were first used outside of the iterative relaxation process, to produce daily plans for individuals (Charypar and Nagel (2005)) and households (Meister *et al.* (2005b)), starting from an empty plan and a list of potential activities (an "agenda"). A simpler genetic algorithm, named Planomat is currently used in the replanning stage of MATSim, which modifies activity durations and mode of a given plan (Meister *et al.* (2006, 2005a)).

2.2.2.2 Algorithm

We present here a genetic algorithm designed to optimize, between two traffic flow simulation iterations, the following dimensions of a joint plan:

1. activity duration;
2. joint trip participation;
3. mode choice.

Solution coding and decoding The coding of the solutions is based on the concept of episode, that we define:

Definition 6 (access trip)**.** *an **access trip** for an activity is a sequence of legs, pick-up and drop-off activities, that ends at this activity. The trip is said **joint** if it contains at least one pick-up/drop-off pair, and is said **individual** otherwise.*

Definition 7 (episode)**.** *an **episode** is constituted by an activity and its relative **access trip** .*

The idea behind this notion is that an episode constitutes a "time slot" in the schedule,within which any duration (travel times and pick-up/drop-off durations) can be set deterministically.

A solution is represented by the following values:

1. floating point values for episode durations;
2. boolean values for joint trip participation;
3. ordered list of possible modes for subtour mode choice.

The encoding of the mode allows to let the mode availability computation at the level of the decoding rather than constraining the genetic operators.

The last episode duration of each individual plan is not encoded. In fact, the durations are subject to the constraints:

$$\begin{cases} d_i \geq 0 & \forall i \in p \ \ \forall p \in P \\ \sum_{i \in p} d_i = d_{day} & \forall p \in P \end{cases} \tag{2.7}$$

where:

d_i the duration of episode i
p the set of indices of the episodes in an individual plan Which translate in, for each $p \in P$,
P the joint plan
with $p = p^* \cup \{l\}$:

$$\begin{cases} \sum_{i \in p^*} d_i \leq d_{day} \\ d_l = d_{day} - \sum_{i \in p^*} d_i \end{cases} \tag{2.8}$$

That is, the last activity duration is always fully determined.

The plan synchronisation is not included in the previous formulation. In fact, the different participants must *arrive* at the same time at the pick-up location. Thus, including this constraint in the genetic operators would be incompatible with the usage of time-dependant travel time estimation. The decoding of the chromosome is done in the following way:

1. joint trip participation is interpreted, and the pick-up/drop-off structure is adapted:
2. the first available mode in the ordered list is allocated to each subtour, using the mode availability definition presented in 2.1.5.2;
3. durations are interpreted, using the algorithm described at Algorithm 3. The travel time estimations are based on the travel times observed in the previous traffic flow simulation. This algorithm ensures individual plans synchronisation.

Algorithm 3 Duration decoder algorithm

```
while not all activities are examined do
  for individual ∈ clique do
    examine next episode;
    if access trip is individual then
      set duration of trip with travel time estimation;
      set activity duration to the remaining episode duration;
    else if all participant pickUp access times are known then
      set duration of pick-up access time with travel time estimation;
      set pick-up duration to the lattest participant arrival time;
      set joint trip duration with travel time estimation;
      set drop-off duration to default duration;
      set egress trip duration with travel time estimation;
      set activity duration to the remaining episode duration;
    else
      estimate access time to the pick up and remember it;
    end if
  end for
end while
```

Genetic operators In the classical genetic algorithm, two genetic operators are used: cross-over and mutation. In the cross-over operator, two solutions are taken from the population, a "crossing point" is chosen randomly, and two new offsprings are generated by exchanging the parts coming after the crossing-point. Another early cross-over is the uniform cross-over, where values of each of the two parents are exchanged with probability 0.5.

In the mutation operator, the values of the offsprings of the cross-over are mutated with a probability p_{mut}, usually chosen very low.

With time, all kind of custom genetic operators were designed. In particular, for real-coded genetic algorithms, new cross-over operators taking advantage of the continuous nature of the search space appeared. The general idea of those operators is to sample points around the parents, generating previously non-existing values. For a revue on this kind of operators, we refer the reader to (Herrera *et al.* (2003)).

Among those approaches, so-called GENOCOP system is of interest for us, as it allows to handle any kind of linear inequality constraints (Michalewicz and Janikow (1991, 1996)).

In this system, 3 cross-over operators are used:

1. the *whole arithmetical cross-over* , which generates new points as a linear combination of the two parents. If $\mathbf{x}$ and $\mathbf{y}$ are the parents, the offsprings will be $a\mathbf{x} + (1-a)\mathbf{y}$ and $a\mathbf{y} + (1-a)\mathbf{x}$, $a \in [0, 1]$. The space defined by linear constraints being convex, the offsprings always belong to

the feasible space if the parents belong to it. In the original system, a is a fixed parameter. We use a random number instead.

2. the *simple arithmetical cross-over* , which is similar to the classical single-point cross-over. Given two parents $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{y} = (y_1, \ldots, y_n)$, and a random crossing point $k \in \{1, 2, \ldots, n\}$, the offsprings are given by $(x_1, \ldots, x_k, ax_{k+1} + (1-a)y_{k+1}, \ldots, ax_n + (1-a)y_n)$ and $(y_1, \ldots, y_k, ay_{k+1} + (1-a)x_{k+1}, \ldots, ay_n + (1-a)x_n)$, with a the biggest number in $[0, 1]$ such that the offsprings respect the constraints.
   The usage of the same coefficient for the two crossed values ensures that the mean vector of the children is the same as the one of the parents, which is generally considered as a good practice. However, our tests revealed that, in our case, the performance is higher when the coefficient are allowed to be different.
3. the *single arithmetical cross-over* , which is similar to the classical uniform cross-over: the linear combination is only realised for one of the elements of the solution vector, a being a random number chosen so that the offsprings respect the constraints. In our setting, we examine each real-coded value in a random order, and perform this recombination operation with probability 0.5.

The discrete values (joint trip participation and mode) are crossed using a uniform cross-over.

For the mutation of real-coded values, we also use the mutations defined in GENOCOP:

1. the *uniform mutation* chooses a new value uniformly from the interval $[l, u]$ of possible values ($l = 0$ in our case).
2. the *non-uniform mutation* sets the new value of x_k as:

$$x'_k = \begin{cases} x_k + \Delta(t, u - x_k) & \text{with probability } 0.5 \\ x_k - \Delta(t, x_k - l) & \text{with probability } 0.5 \end{cases} \tag{2.9}$$

where:

$\Delta(t, y) =$	is a random number which probability to be close to x_k increases
$y(1 - r^{(1-\frac{t}{T})b})$	with time
r	is a random number drawn uniformly from $[0, 1]$
t	is the number of the current iteration
T	is the maximum number of iterations
b	is a model parameter, controlling the degree of non-uniformity

The mutation of a boolean value x is done by setting it to $\neg x$, the mutation of a list value is done by shuffling its elements.

Selection scheme A well known problem of genetic algorithms (and of all heuristic and metaheuristic methods in general) is the *premature convergence* problem. A good convergence behaviour is often understood as a good balance between *diversification* and *intensification* (also called *exploration* and *exploitation*): pure diversification would be a simple random search, which may not exploit the structure

of the search space; pure intensification would be an hill-climbing algorithm, which has great chances to get trapped in a local optimum.

In our case in particular, unsynchronized solutions (no joint trip) constitute local optima, much easier to reach than the possible synchronized optimum. Thus, diversification (or diversity preservation) has to be favoured in some way.

In a genetic algorithm, the selection scheme is of course fundamental to keep diversity. In the case of multimodal functions, one feels that diversity should lead to having sub-populations exploring different promising areas. This is with this goal that so-called *niching* selection schemes were developed. One way to achieve this niching is called *fitness sharing* (Holland (1975); Horn (1997)), were solutions which genotypes are close according to some metric have to "share" fitness. Those methods are however expensive, as they require numerous distance calculations.

Among less expensive techniques belongs restricted tournament selection (RTS) (Harik (1995)).

Its principle is the following: for each newly generated solution, w solutions from the previous generation are selected randomly for competition. From those solutions, the fitness of the one that is closest to the newly generated offspring is compared to the fitness of the new solution. The solution with the highest fitness is kept.

With this setting, newly generated solutions tend to replace close solutions, even if far solutions have a very bad fitness. Harik shows that this selection schemes preserves multimodal solutions with high probability if w is greater than the number of local optima.

This selection scheme was developed for a steady state genetic algorithm, where newly created offsprings directly compete with the current population. To use it in a generational genetic algorithm, where a set of new values is generated and then integrated, we also consider new offsprings as part of the population once integrated. That is, a newly generated chromosome can be inserted in the population and then replaced in the same generation.

The metric retained here uses the 1-distance for real-coded values, the scaled Hamming distance (1 if the two values are identical, 0 otherwise) for boolean values, and nothing for the list values (*i.e.* we do not consider the mode in the distance):

$$d(\mathbf{x}, \mathbf{y}) = \sum_{i \in \mathcal{R}} |x_i - y_i| + \sigma \sum_{i \in \mathcal{B}} \mathbb{1}(x_i \neq y_i) \tag{2.10}$$

where:

- $\mathcal{R}$ is the set of indices in the chromosome corresponding to real values
- $\mathcal{B}$ is the set of indices in the chromosome corresponding to boolean values

Stopping criterion As pointed out in 2.2.1, plans are optimized for groups of agents which plans are interdependent. Thus, size of instances may vary much more than when replanning individuals.

Moreover, the complexity of the problem typically changes with the number of individuals in the group. Indeed agents are grouped because potential joint trips were identified for them, thus requiring the algorithm to be able to find promising collaboration schemes.

In fact, for individual plans, the mere random generation of plans is sufficient to obtain good solutions, while for joint plans with possible joint trips, joint trips are only found with good parameters and after a sufficient number of iterations. Figures 2.1-2.3 give an idea of this fact.

Thus, a fitness monitoring process was introduced, to avoid passing too much time optimizing easy instances.

It works in the following way: after a given number of generations n_{start}, the fitness of the best found solution is remembered. Then, every n_{period} generations, the new best fitness is measured: if the improvement since the last monitoring is less than a given amount Δ_{min}, the algorithm is stopped and the best found solution is returned.

2.2.3 Parameters

2.2.3.1 Setting the parameters

The parameters of the algorithm, as well as their description and assigned values, are presented in Table 2.1.

As can be seen, our algorithm has quite an important number of parameters. Thus, while parameter setting is known to be critical for the performance of a genetic algorithm, finding appropriate values is non-trivial.

This problem of the importance of parameter values is well known since the beginning of the genetic algorithm history.

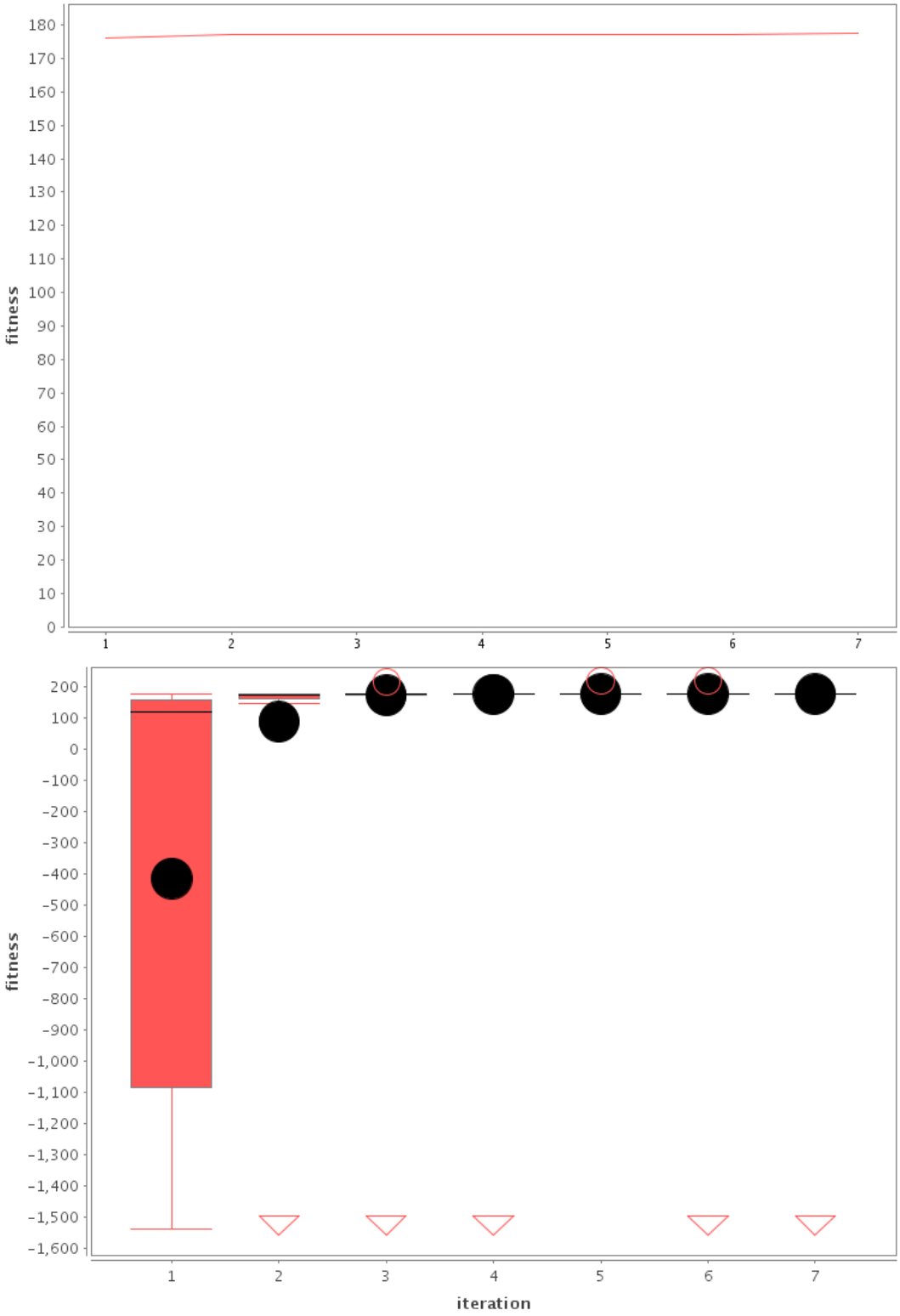
The first approach to tackle this problem has been to try to find good "default" values, which would be suitable for most genetic algorithms.

However, it is know widely acknowledged that the "goodness" of parameters depend on the problem coding, the particular instance, and may even change during the evolution process.

Thus, two approaches have been proposed to tackle this issue:

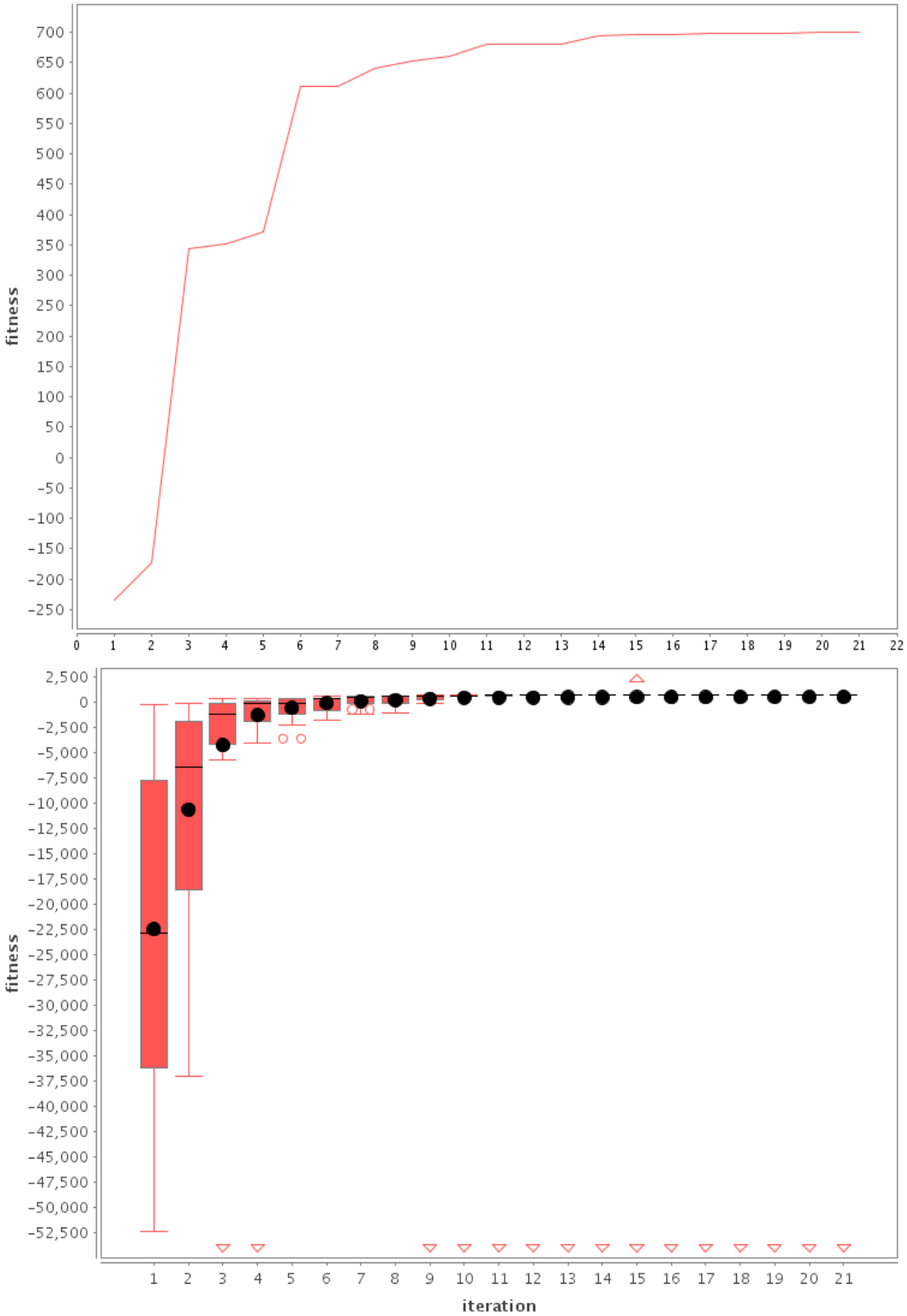
1. *parameter control*, also sometimes called "parameter-less" genetic algorithm, where parameter values change during the evolution process. This may take the form of *parameter evolution*, where the parameters of the algorithm are part of the chromosome and co-evolve with solutions, or of an

Figure 2.1: Evolution of the fitness for a clique of one individual: maximum fitness and fitness distribution



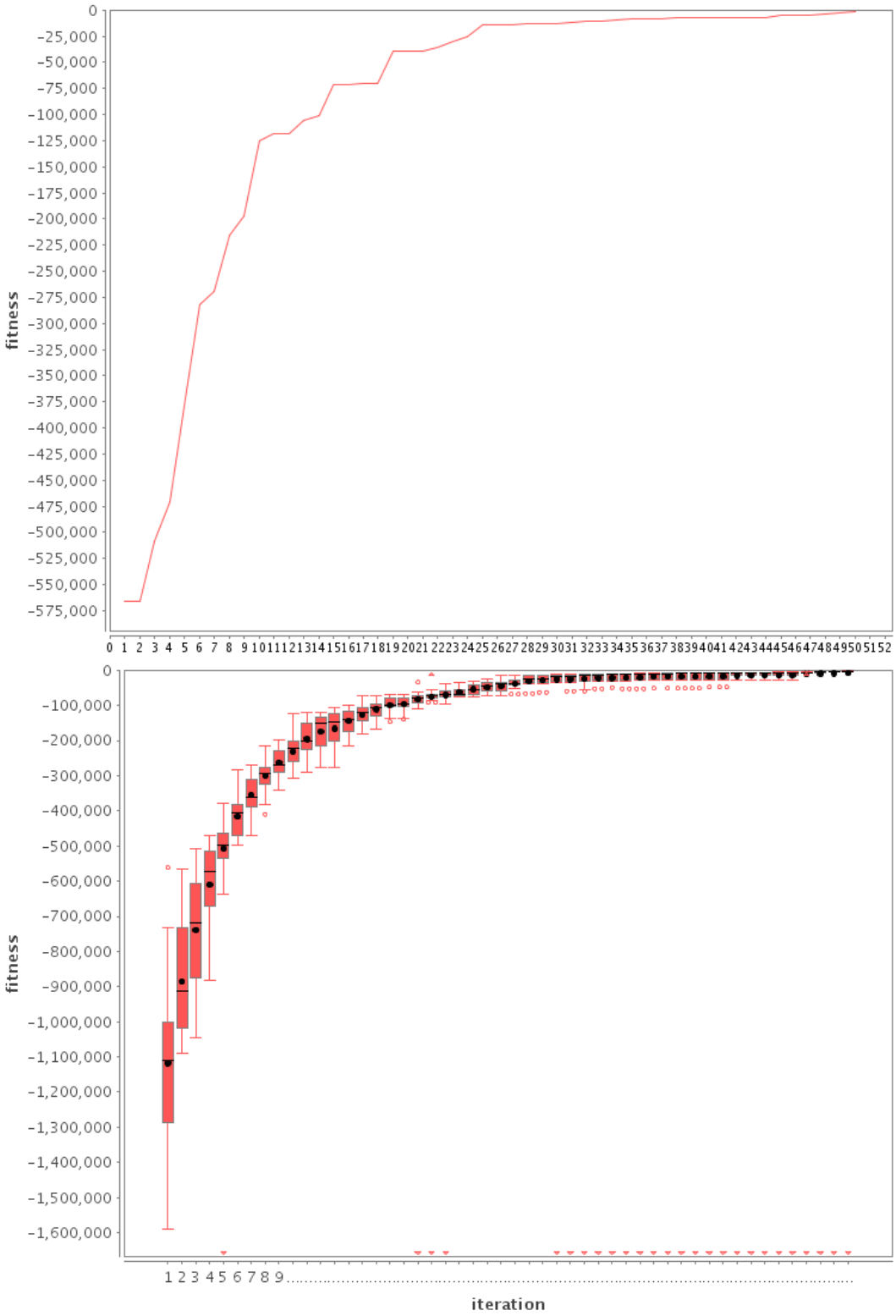
a priori fixed evolution scheme. The parameters that are evolved may consist only of the genetic operators probabilities, or include other parameters, including population size (Harik and Lobo (1999)). The interested reader may want to consult (Eiben *et al.* (1999)) for a classification of those approaches.

Figure 2.2: Evolution of the fitness for a clique of three individuals: maximum fitness and fitness distribution



2. *parameter tuning*, where parameters are optimised *a priori* by an optimisation algorithm, the fitness of a parameter set being given by some performance metric on one or several test instances. A genetic algorithm is often used, being than called "meta genetic algorithm " (Grefenstette (1986)).

Figure 2.3: Evolution of the fitness for a clique of forty individuals: maximum fitness and fitness distribution



We chose here to use the *parameter tuning* approach, for the following reasons:

1. our problem typically requires very few generations, compared to other problems. Thus, it is not obvious that adaptive parameters would improve something.

2. there exists as many parameter control techniques as implementations of it, and no general guidelines can be found to implement such techniques. Thus, finding the good parameter control approach for an application may require a significant amount of extra care.
3. the *parameter control* approaches require more computational power, and our application is quite performance critical (it is used in the inner loops of MATSim).

2.2.3.2 The parameter optimisation algorithm

To optimise the parameters of our algorithm, a standard generational genetic algorithm was used. The parameters of this meta genetic algorithm are summarized in Table 2.2.

The aim of using an optimisation algorithm to tune the parameters values is to find the parameters with the best trade-off between the ability of the algorithm to find a near-optimal solution and the computational time required to converge.

Thus, the following expression was retained for the fitness of a parameter set:

$$F(\mathbf{x}) = \sum_{i \in \mathcal{T}} \sum_{j \in \mathcal{R}_i} \left(f_i^{(j)}(\mathbf{x}) - \alpha \, t_i^{(j)}(\mathbf{x}) \right) \tag{2.11}$$

where:

$\mathbf{x}$	is the parameter values to score.
$\mathcal{T}$	is the set of indices of the test instances.
$\mathcal{R}_i$	is the set of indices of the random seeds for instance i.
$f_i^{(j)}(\mathbf{x})$	is the average fitness per clique member for the found optimum, with algorithm parameters $\mathbf{x}$, for instance i, random seed j.
α	is a parameter controlling the relative importance of speed of convergence and solution quality. This should not be interpreted as the wanted "speed" of the algorithm: typically, low values work quite well, by penalising the slower of two good solutions, without giving too much advantage to bad but fast solutions.
$t_i^{(j)}(\mathbf{x})$	is the CPU time per clique member needed to converge with algorithm parameters $\mathbf{x}$, for instance i, random seed j.

The optimisation procedure is the following:

1. run the traffic flow simulation on the scenario to simulate, to get travel time estimates. Using the scenario to run for the parameter optimisation allows to tune the parameters for the particular instance at hand.
2. choose n_{plans} joint plans randomly from the population. The mere random generation of solution being able to find good individual plans, and evaluating the parameters on large instances being expensive, only cliques between 3 and 5 agents were drawn.

3. run the meta genetic algorithm with those test instances.
4. return optimized parameters and exit.

The resulting values are presented in Table 2.1. Those value lead to solution scores as good as with intuitively fixed parameters, but in much less time.

Table 2.1: Algorithm parameters

parameter	description	range	value	
N_{chrom}	population size: number of solutions maintained by the genetic algorithm	$\{2, \ldots, 100\}$	28	*
$\mathbb{1}_{inPlace}$	1 if the mutation is "in place" (classical genetic algorithm), 0 otherwise	$\{0, 1\}$	0	*
p_{mut}	mutation probability: if $\mathbb{1}_{inPlace} = 1$, each gene in the offsprings of the cross-over operators is mutated with this probability. If $\mathbb{1}_{inPlace} = 0$, each gene of every chromosome in the previous generation is mutated with this probability. In this setting, the unmutated version of the chromosome is also candidate for selection.	$[0, 1]$	0.06	*
p_{nunif}	probability for a real-valued gene, given that it will be mutated, to be mutated by the non-uniform mutation operator	$[0, 1]$	0.04	*
p_{whole}	offspring rate for the whole arithmetical cross-over operator: $\lceil p_{whole} \times N_{chrom} \rceil$ couples of chromosomes are operated	$[0, 1]$	0.36	*
p_{single}	offspring rate for the single arithmetical cross-over operator: $\lceil p_{single} \times N_{chrom} \rceil$ couples of chromosomes are operated	$[0, 1]$	0.40	*
p_{simple}	offspring rate for the simple arithmetical cross-over operator: $\lceil p_{simple} \times N_{chrom} \rceil$ couples of chromosomes are operated	$[0, 1]$	0.84	*
b	degree of non-uniformity of the non-uniform mutation	$[0, 50]$	34.60	*
w	restricted tournament selection window size	$[2, 20]$	6	*
σ	scaling factor of the discrete distance for the restricted tournament selection	$[0, 10^7]$	1407972	*
n_{start}	number of generation before first fitness monitoring		3	
n_{period}	monitoring period		1	
Δ_{min}	minimum fitness improvement per agent in the clique (€)		0.075	
n_{max}	maximum number of generation before return		50	

* those values were determined by an algorithm. See 2.2.3 for details. The searched space is indicated in the "range" column.

Table 2.2: Parameters of the meta genetic algorithm

parameter	description	value
p_{mut}	probability for a gene to be mutated. In this setting, the chromosome to mutate is taken from the previous generation, and is added to the candidate chromosomes if mutation occur.	0.1
p_{co}	Cross-over rate	0.6
N_{chrom}	Population size	40
τ_{orig}	Original rate: at each generation, the $\lceil \tau_{orig} N_{chrom} \rceil$ best solutions are selected for the next generation, and the population is filled with new random instances.	0.8
n_{plans}	Number of test instances	10
n_{runs}	Number of runs of the algorithm on each instance, with different random seeds.	1
α	Fitness "speed" parameter (see equation (2.11))	1 €.s^{-1}

3 Simulation experiments

3.1 Simulation setting

3.1.1 Scenario

As stated above, the proposed method relies on *a priori* identified joint trips possibilities.

For those test runs, it was chosen to use car-pooling matings computed by PTV SWISS AG, Bern, in a joint project with IVT.

Those matings were computed from the steady state of a MATSim simulation without joint trips, using time windows and a maximum detour time for the driver. The driver is assumed to pick up the passenger at its departure location, and drop him at his arrival location, during one of its trips. This approach in terms of detours and admissible time windows is usual in operational car-pooling algorithms, and the formulation with pre-identified drivers and passengers is sometimes referred to as the car-pooling problem (Buchholz (1997); Baldacci *et al.* (2004)). Mode consistency is enforced at the subtour level.

This procedure results in cliques with up to 106 agents, but with a vast majority of cliques of low cardinality.

The affectation procedure was run on a synthetic population containing as many agents as the simulated zone. However, it is usual to simulate only a sample population (Balmer *et al.* (2006)) to reduce computationnal cost. In order to test the robustess of our module facing cliques of high cardinality, a 10% sample was drawn, stratified by clique size. That is, the proportion of agents pertaining to cliques of each size is enforced to be the same in the sample and in the initial population, rounding to the greater integer number of cliques. The resulting clique size distribution can be seen in Figure 3.1.

3.1.2 MATSim set-up

3.1.2.1 Strategies

Between each traffic flow simulation, so-called strategies are executed on random agents (corresponding here to cliques). For each agent, a strategy is selected randomly according to specified probabilities. Those strategies and their probabilities are taken from the work of Meister *et al.* (2010), and are presented in Table 3.1.

Figure 3.1: Clique size distribution in the 10% sample

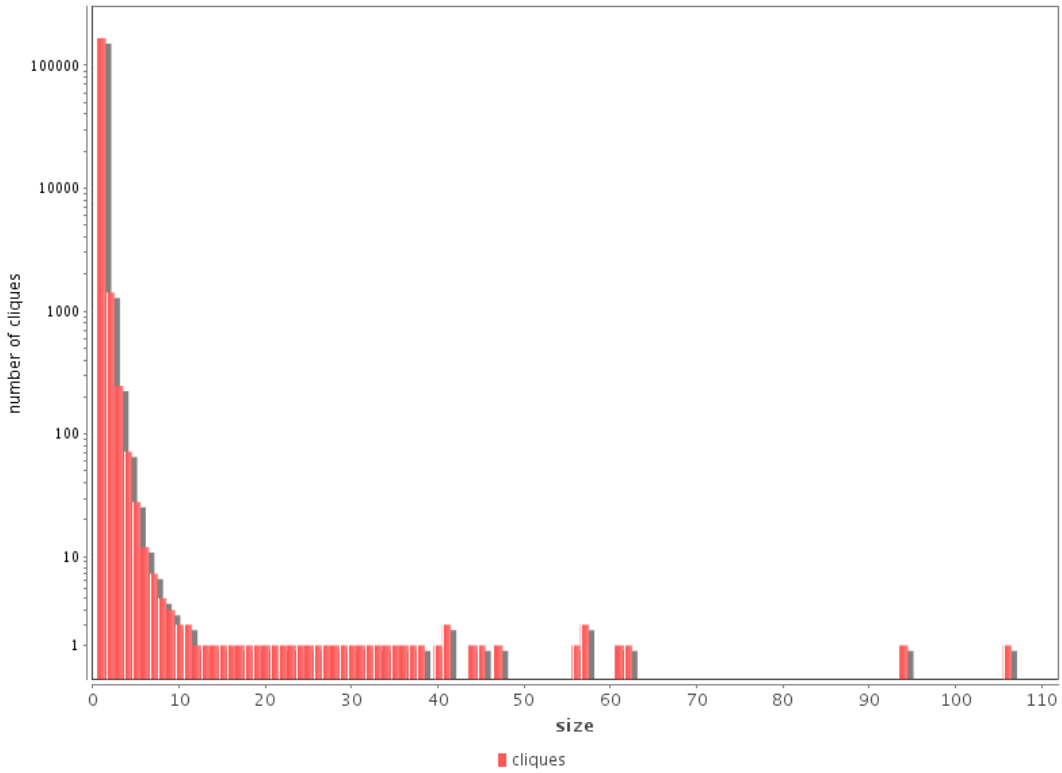


Table 3.1: Strategies used between the runs

Strategy	Description	Probability
ReRoute	Least cost re-routing	0.1
ExpPlanBeta	Selection of a plan from the agent data base, with a logit-like probability: a plan i with score S_i is selected with probability $\frac{e^{\beta S_i}}{\sum_j e^{\beta S_j}}$. Here, $\beta = 2$	0.8
Replanning	Clique replanning genetic algorithm	0.1

3.1.2.2 Utility parameters

A good specification of the utility is a key point to obtain realistic results. Those parameters can be calibrated, either by hand (Meister *et al.* (2010)) or automatically (Flötteröd *et al.* (2010a,b)) to minimise the distance between the simulation output and some control totals, as for example the mode shares.

For the algorithm performance analysis, it was however chosen to use the default MATSim values (Meister *et al.* (2006)). Pick-up and drop-off activities have the utility of waiting.

Those values are presented in Table 3.2.

Table 3.2: Values of the utility parameters

parameter (see 1.1.1.2)	value
β_{dur}	6 € $.h^{-1}$
β_{trav}	-6 € $.h^{-1}$
β_{wait}	0 € $.h^{-1}$
$\beta_{late.ar}$	-18 € $.h^{-1}$
$\beta_{early.dp}$	0 € $.h^{-1}$
$\beta_{short.dur}$	0 € $.h^{-1}$

3.1.2.3 Time-dependent travel time estimation

As stated in 2.2.2.2, the algorithm uses time-dependant travel time estimates.

Here are some remarks on the way those estimates are computed:

1. for performance reasons, the estimation use fixed routes for every origin/destination pair: if this origin/destination existed in the plan to modify, the corresponding route is taken; otherwise, a least-cost routing algorithm is used, and the route is remembered.
   Thus, the routing does not includes traffic jam avoidance at this stage.
2. for non-car modes, due to implentation reasons, it is currently impossible to get travel time estimates without actually running a routing algorithm. Thus, for such modes, the travel time estimate is not time-dependant.
   Thus, the travel time at noon is used for such modes.

3.2 Simulation experiments

To test the performance of the algorithm, two comparative runs were performed: first, the performance of the proposed algorithm on individual plans is compared with the performance of Planomat, the current genetic algorithm based replanning module. It shows that the proposed algorithm is competitive with the current module, when no joint trips are optimised. Then, an extensive analysis of the behaviour of the algorithm is proposed. It is seen that if the algorithm is able to identify relevant joint trips, the resulting utilities are lower than in the case where no joint trips are optimised. This behaviour is analised as a lack of exploitation behaviour when joint trips are optimised, and may be solved with minor changes in the optimisation settings.

3.2.1 Joint Replanning Algorithm vs. Planomat

The proposed module, in the case where no joint trips are identified, optimises the same dimensions as the current replanning module called Planomat.

Thus, we use a MATSim run with Planomat, on the same scenario, as a "base case" to analyse the behaviour of our module in the case where no joint trip is identified.

Figure 3.2: Score evolution when using Planomat for replanning

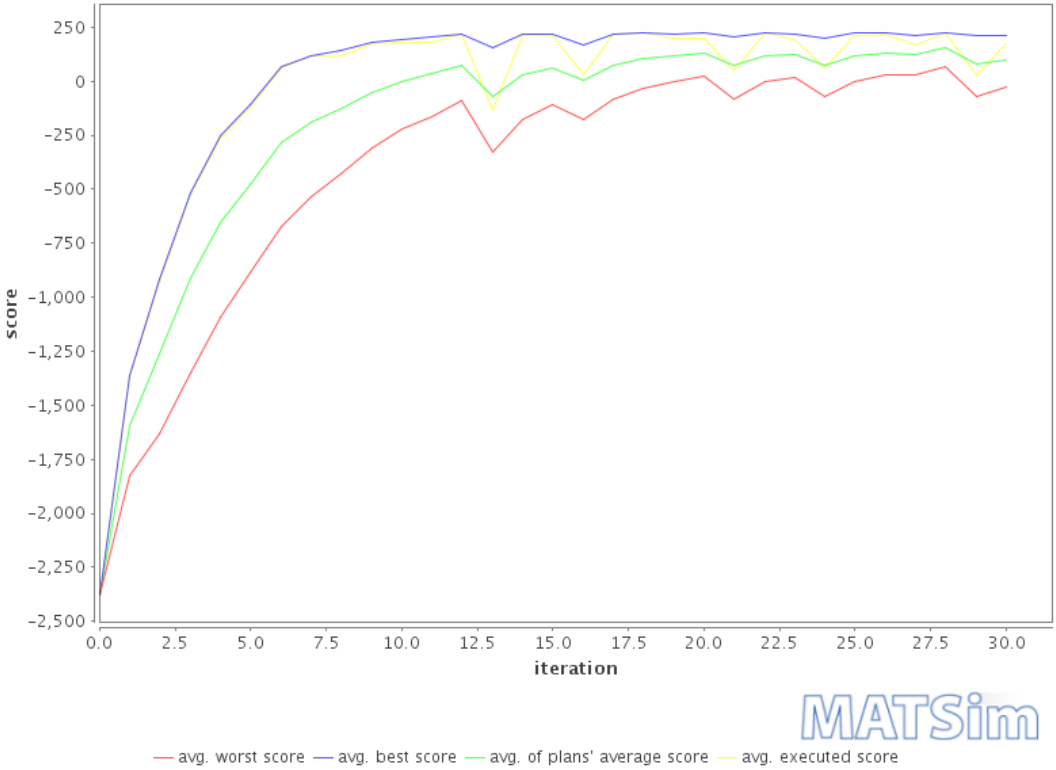
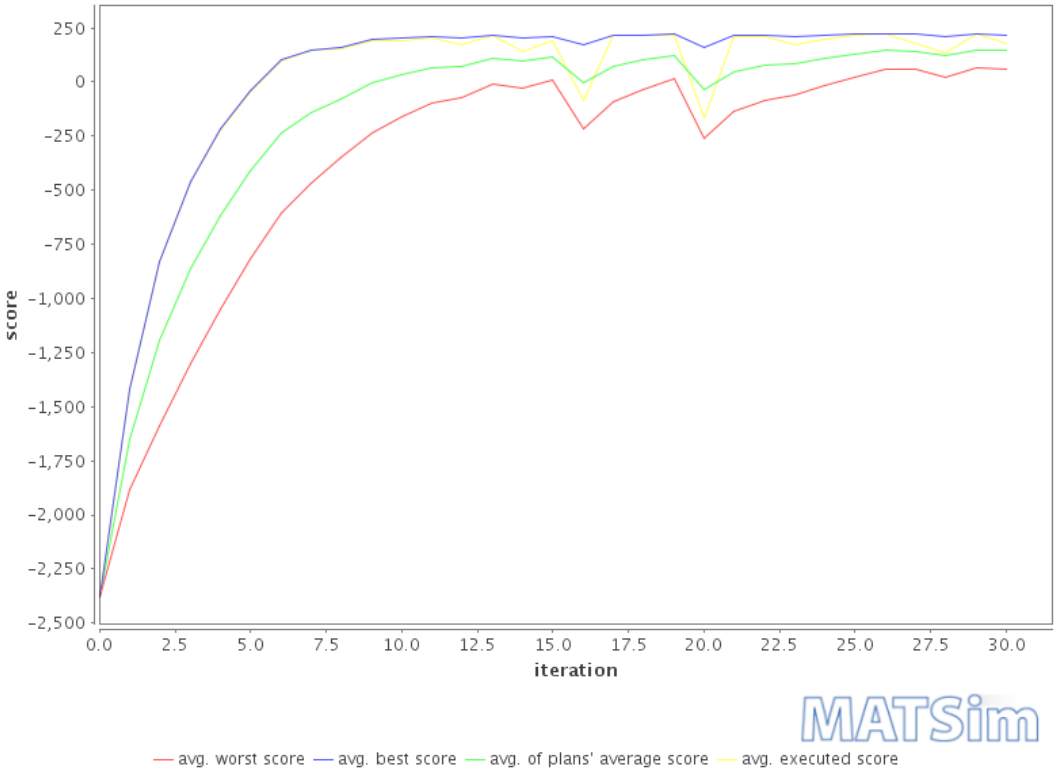


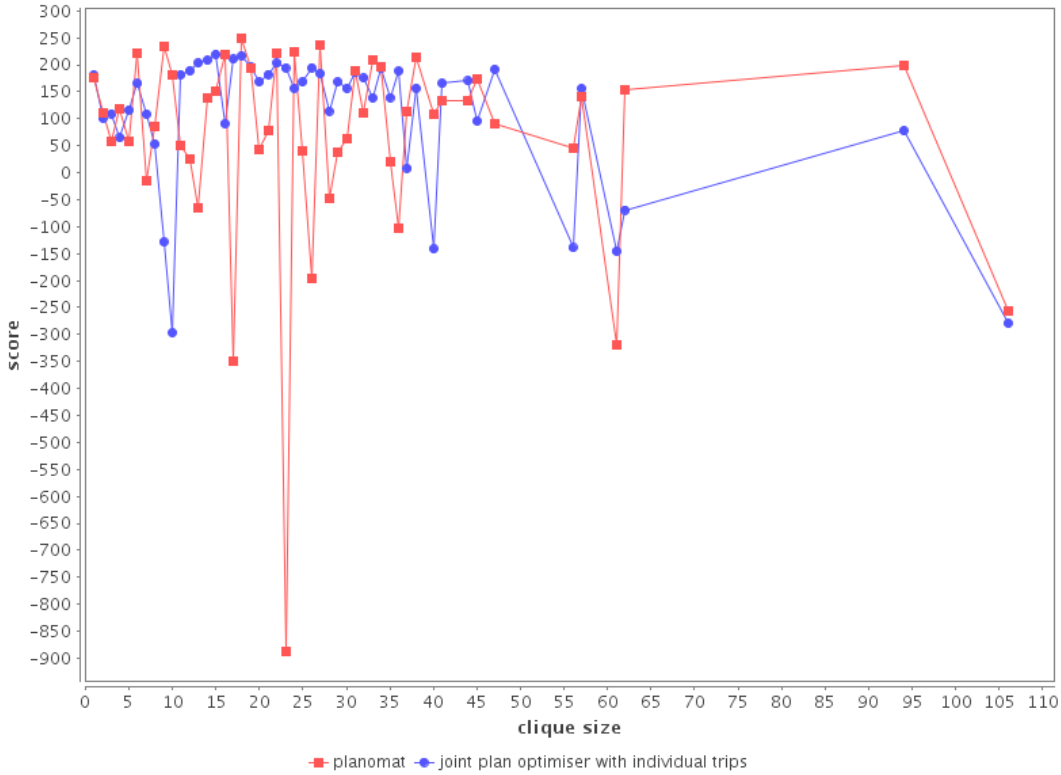
Figure 3.3: Score evolution when using the joint plan optimisation algorithm for replanning



Figures 3.2 and 3.3 show a very similar convergence behaviour for both replanning algorithms.

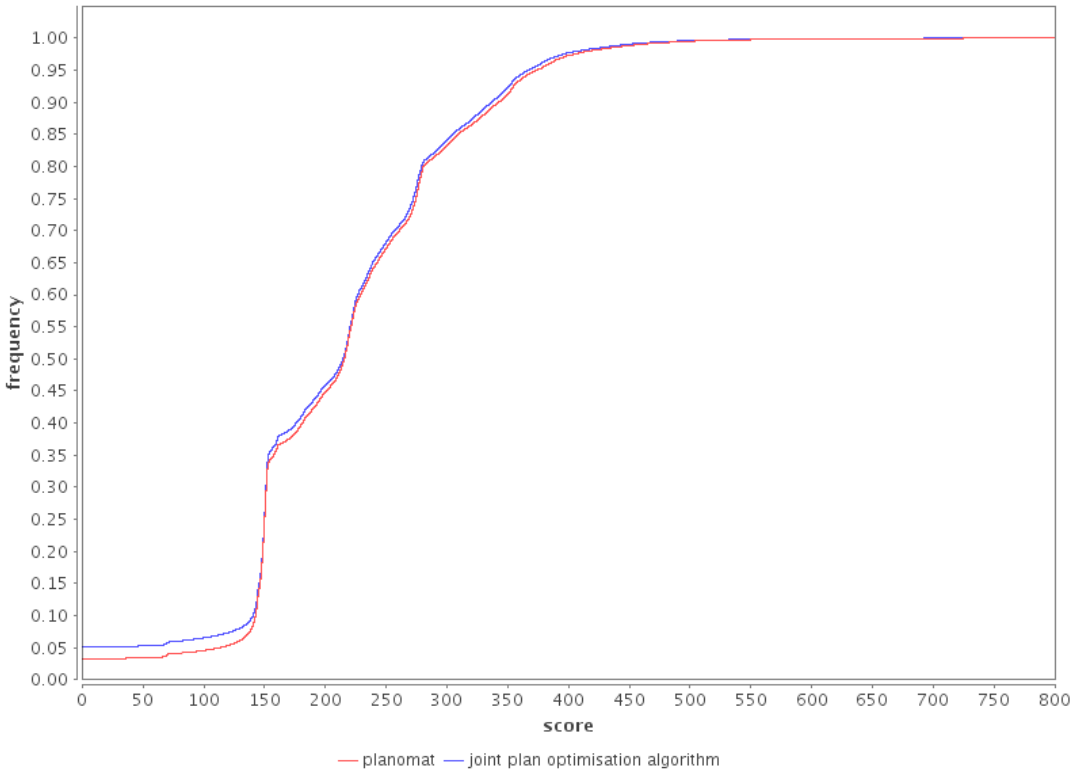
However, as the joint plan optimisation algorithm works on cliques, it may have more difficulties to optimise an individual plan when the agent belongs to a large clique. Figure 3.5 shows the average score for the agents with the two replanning modules, as a function of the size of the clique the agent belongs to. The more chaotic aspect of Planomat's scores is probably due to the slightly higher replanning rate implied by the group level strategy selection: maybe all agents of a clique of large size were not replanned by Planomat. However, the scores seem quite similar.

Figure 3.4: Average executed score at the steady state as a function of the clique size: comparison with Planomat



At a more aggregated level, the produced score distributions are very similar, even though the clique replanning produces more plans of low score.

Figure 3.5: Cumulative distribution of the scores: Planomat and joint plan optimisation algorithm



3.2.2 Behaviour of the algorithm

3.2.2.1 Consistency of the results

As stated in 2.2.2.2, a major risk for an genetic algorithm is premature convergence towards a local optimum. In our case, the plans without joint trips constitute a local optimum, easier to reach than the global synchronised optimum, if it exists.

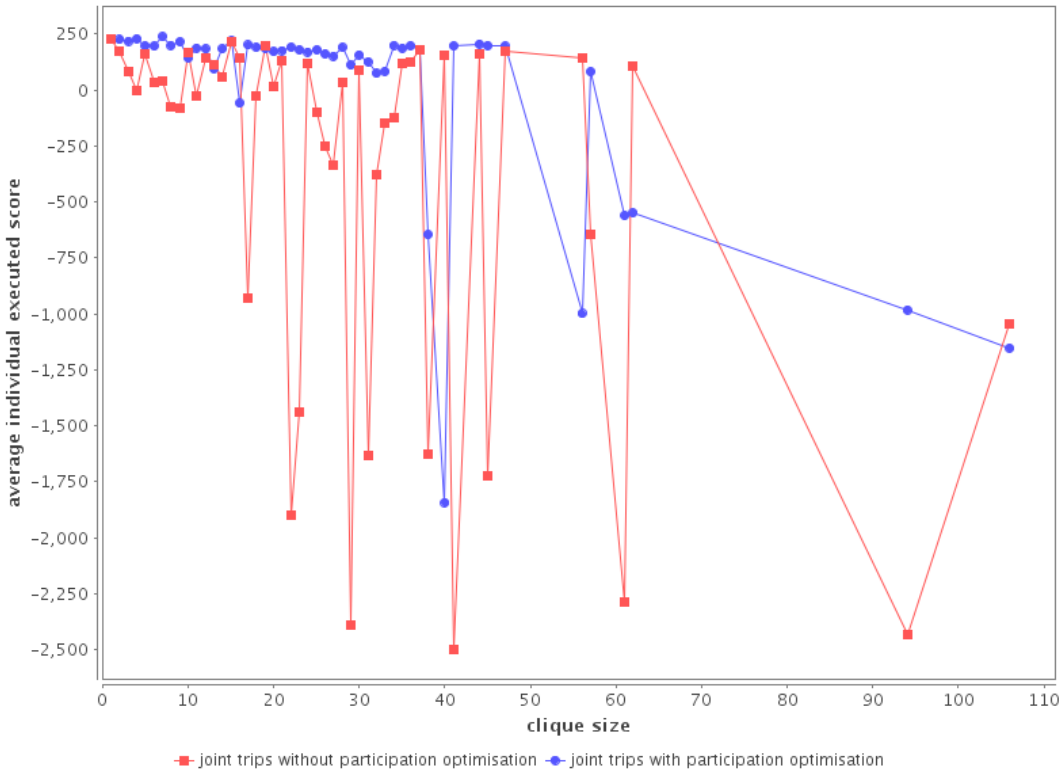
Such a premature convergence could lead to a higher score when synchronisation is enforced. Figure 3.6 represents the average executed score as a function of the clique size, for the two settings, at the steady state.

The figure may look quite chaotic, with some really low scores. Those low scores may result from two phenomena:

1. for high cardinality cliques, generally only one clique of each size exists. Thus, some low scores may correspond to non-replanned cliques.
2. the results without joint trip participation optimisation seem much more chaotic than the ones with joint trip participation optimisation. This may come either from expensive affected joint trips or from a difficulty of the algorithm to find optimal plans with a high number of joint trips.

However, the figure shows the expected behaviour, with higher scores when joint trip participation is optimised.

Figure 3.6: Average executed score at the steady state as a function of the clique size: impact of joint trip optimisation



Figures 3.7 and 3.8 show in more details the individual utility gains, depending on the size of the clique the agent belongs to, when passing from no joint trip participation optimisation to joint trip participation optimisation. Interestingly, very few agents see their score decreasing when joint trip participation is not enforced, letting think in a good convergence behaviour.

Another inconsistency may occur due to the number of parameters in a joint plan: the scores of the plans without joint trips may be higher than the ones of the same plans with optimised joint trips.

Figure 3.9 represent the average executed score as a function of the clique size, for plans with and without possible joint trips, at the steady state. The plans without joint trips were optimised using the joint replanning module, with the same cliques as for the optimisation with joint trips.

For cliques up to 15 members, the plans with joint trips have better scores than the plans without. The joint trip optimisation seems even to make scores more uniform, as it "corrects" the lower average scores in the low size cliques, which were also observed with Planomat. However, joint plan optimisation for large cliques fails, giving scores much lower than the ones without joint trips.

Figures 3.10 and 3.11 show that if approximately half of the cliques see their utility decreasing when joint trips are optimised (median of the gains close to zero), the utility gains are much higher than the losses. Moreover, the most important losses only occur for cliques of high cardinality (more than 30 members). However, the median loss increases with the average number of joint trips per individual, and the median is never strongly higher to 0, indicating a slight difficulty to converge when joint trips are allowed.

Figure 3.7: Individual utility gains when joint trip participation is optimised

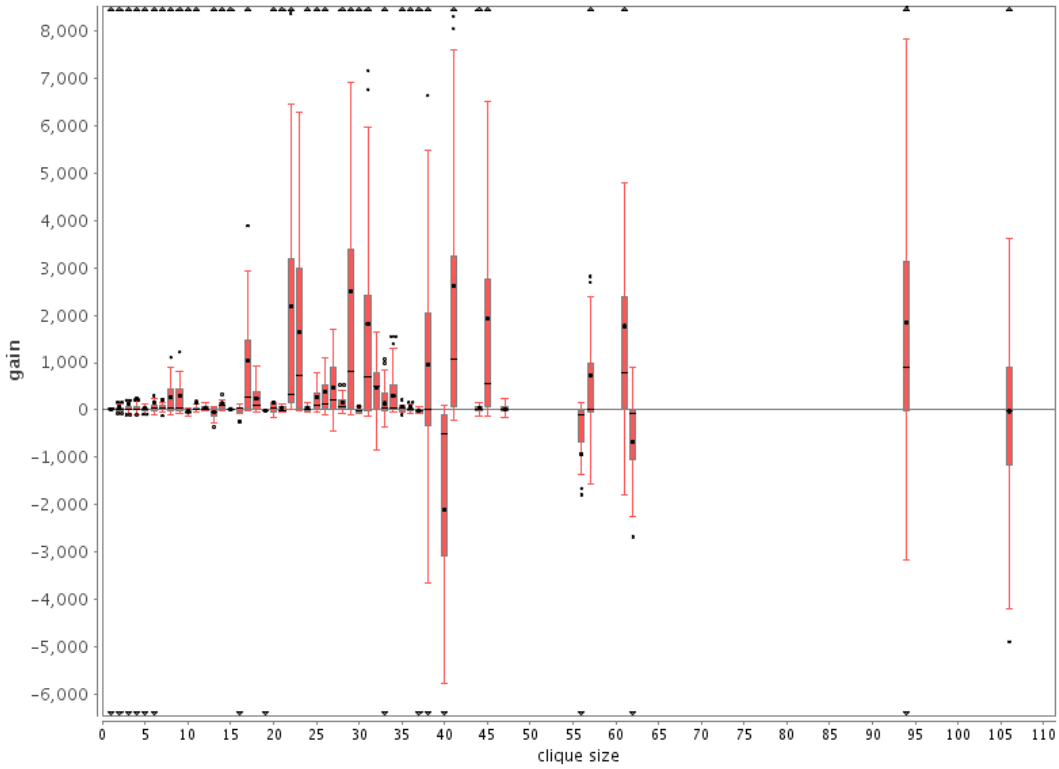


Figure 3.8: Individual utility gains when joint trip participation is optimised - cliques of less than 30 members only

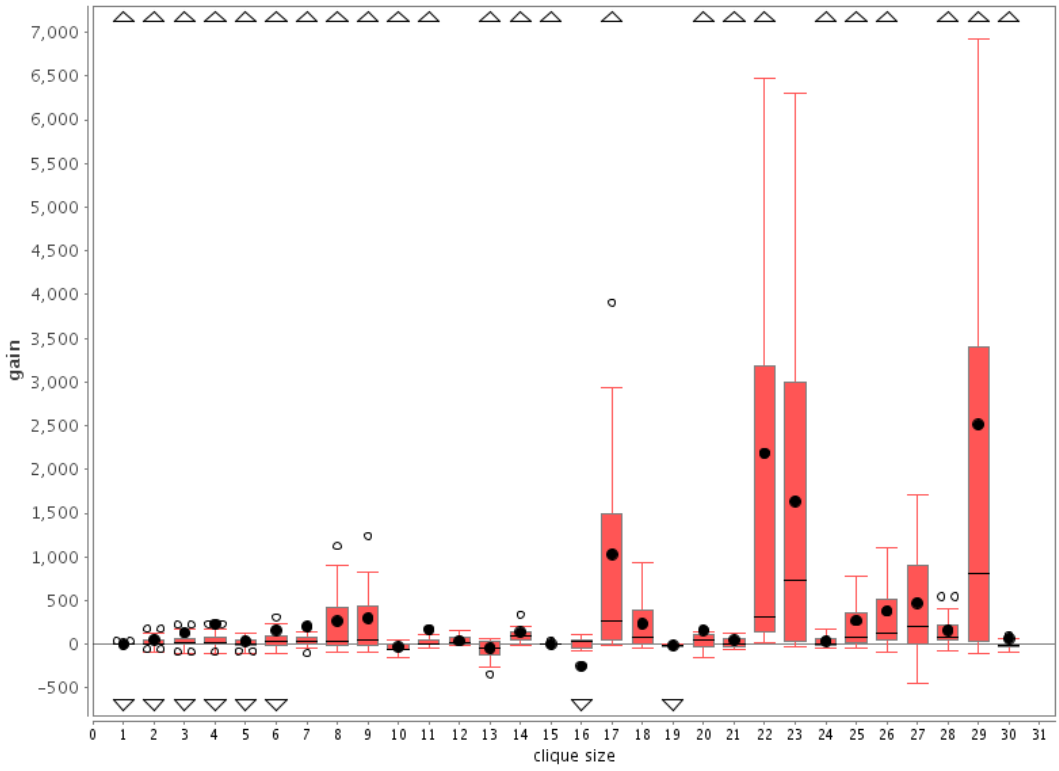


Figure 3.9: Average executed score at the steady state as a function of the clique size: individual trips

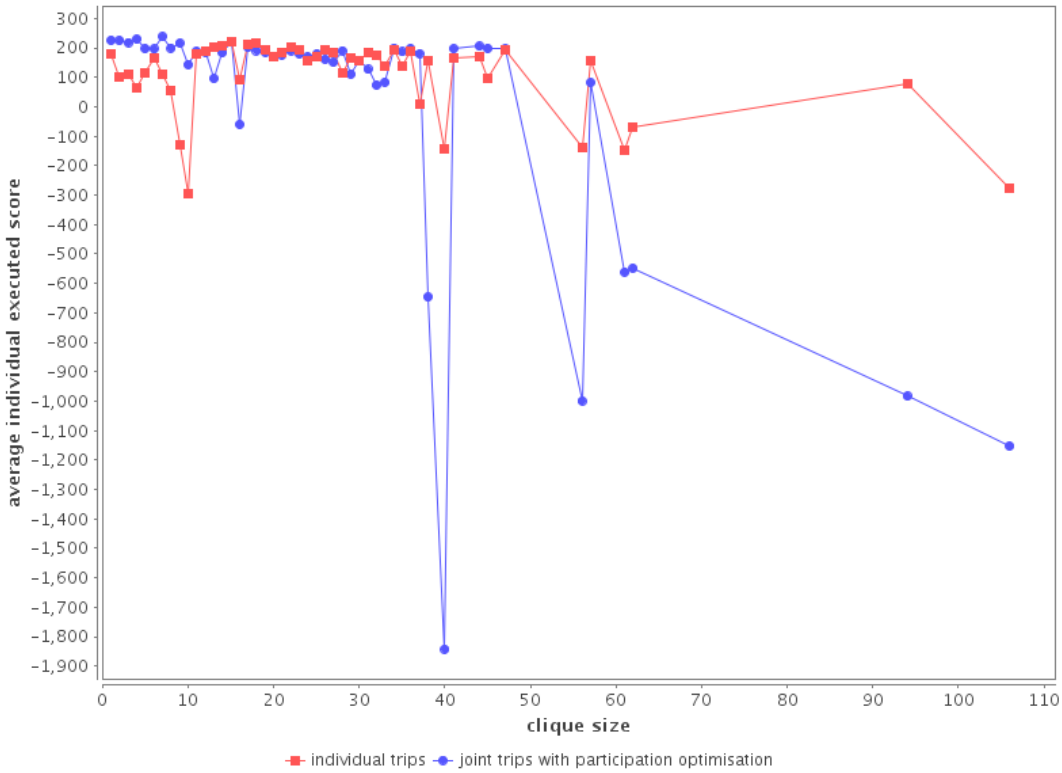
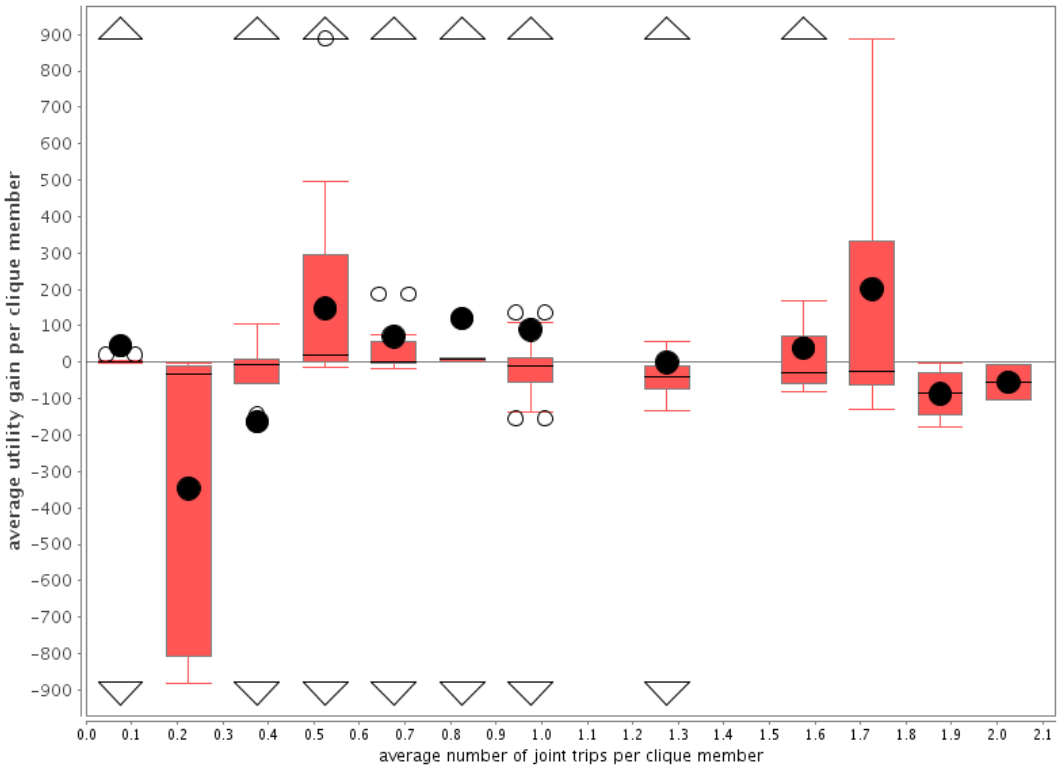
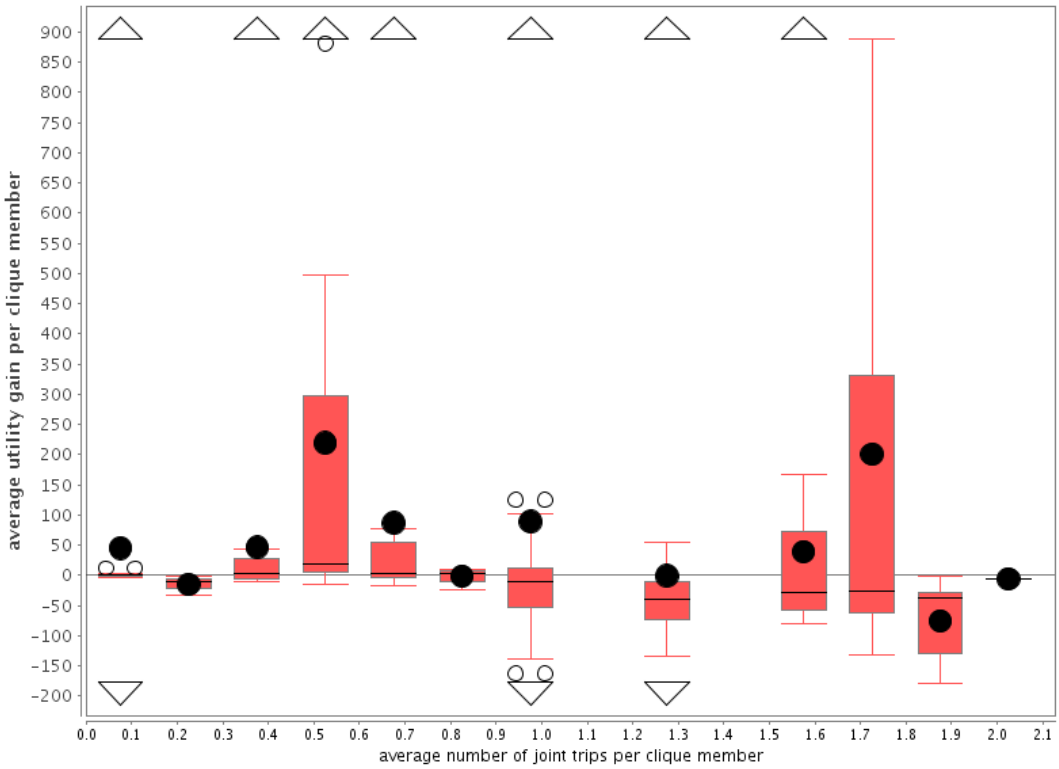


Figure 3.10: Clique utility gains when passing from plans without joint trips to plan with joint trips



Moreover, when looking at the distribution of the utilities at the individual level, one notices that optimising plans without joint trips possibilities gives a greater proportion of good plans than when

Figure 3.11: Clique utility gains when passing from plans without joint trips to plan with joint trips - cliques of less than 30 members only



optimising with joint trip possibilities, even when focussing on the individuals for which joint trips where retained during the optimisation procedure (Figure 3.12). This fact seems to corroborate the previously pointed convergence problem.

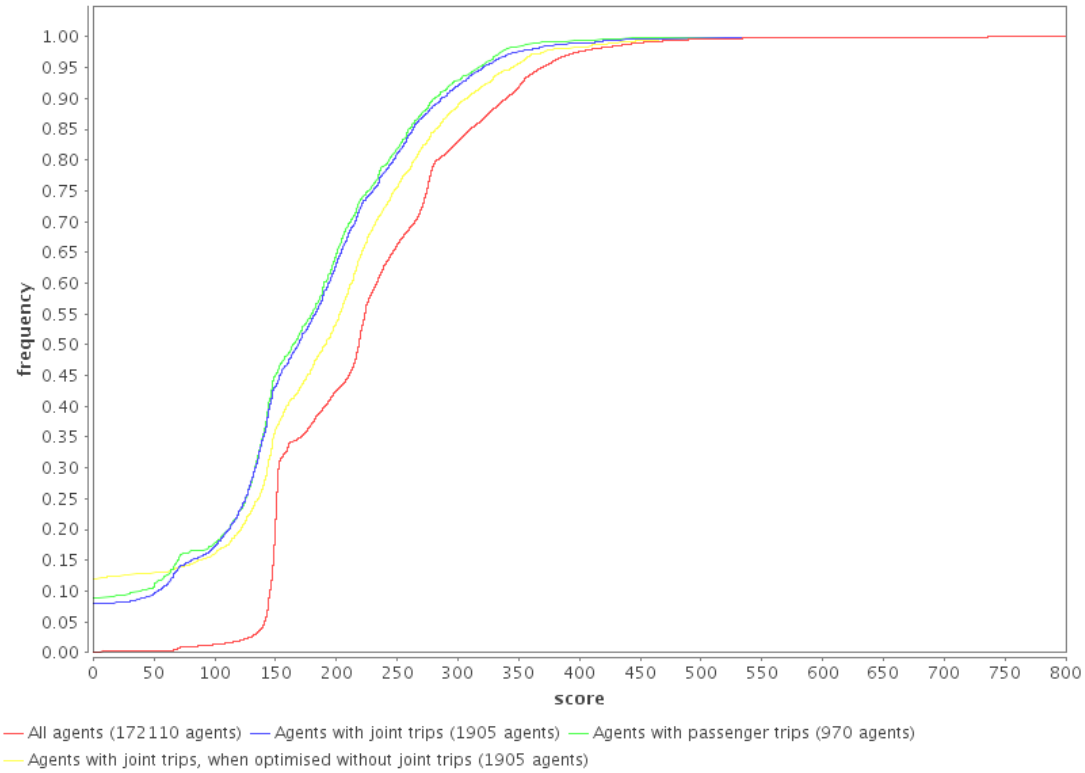
In fact, one may remember that in the chosen selection scheme, new solutions only replace close solutions (see 2.2.2.2), and that the algorithm parameters make the differences in the selected joint trips much more important than the differences of durations (see Table 2.1). This selection scheme was introduced to avoid an observed premature convergence towards unsynchronised solutions; however, due to the small population size, niching may result in an insufficient number of chromosomes exploring each promising part of the search space, when such promising areas are numerous, leading to more exploration and less exploitation. Several improvements may solve this problem, as using an hill-climbing algorithm on the best found solution or using problem-dependent parameters (for example, varying with the number of joint trips in the plan).

Finally, we observe that from some clique size (around 50 agents), the algorithm fails to find good scores for both settings.

3.2.2.2 Robustness of the algorithm facing large scale problems

Another problem that could occur is a difficulty to optimise large scale instances, due to the explosion of the search space size.

Figure 3.12: Distribution of the scores for different categories of agents, with and without joint trips optimisation



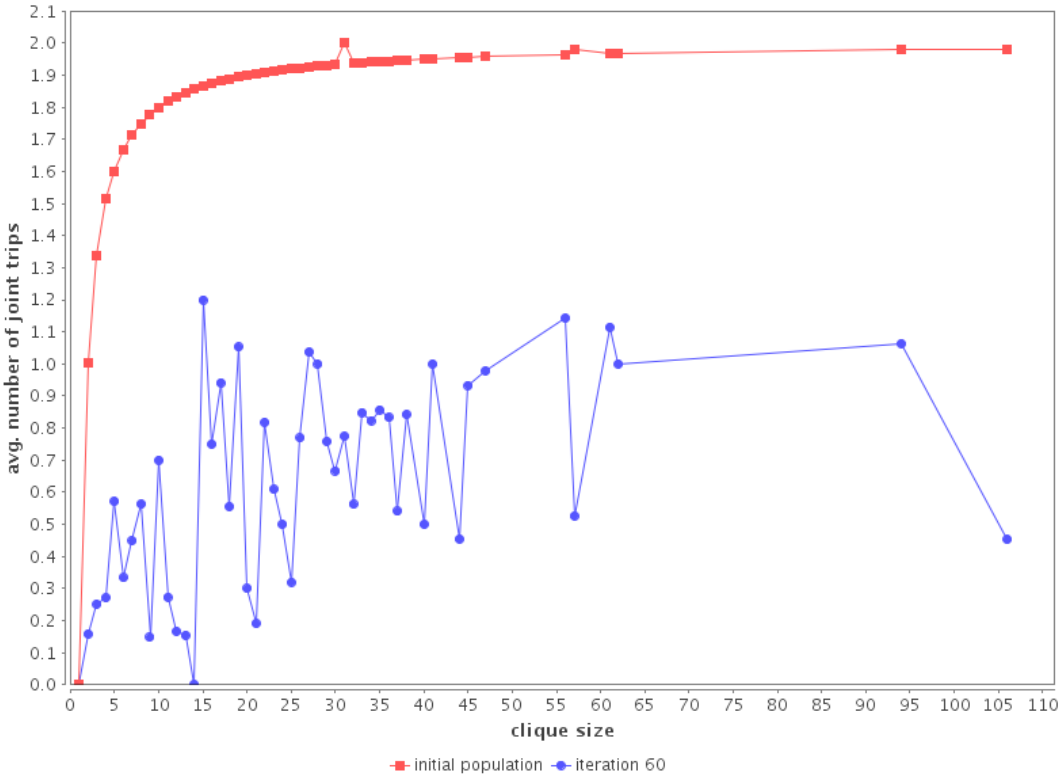
Such a problem would have two visible effects:

- a decrease in the score of the optimised plans with the clique size;
- a lower portion of joint trips "accepted" during the optimisation.

Figure 3.6 and Figure 3.9 show that optimised scores remain approximately constant with the clique size, at least for cliques of less than 50 agents. However, for high clique size, the optimised scores are very low, and giving joint trips possibilities just makes the convergence harder and thus the found "optimum" even worst.

On Figure 3.13, no decrease of the average number of joint trips per individual is visible when the clique size is increasing.

Figure 3.13: Average number of joint trips per individual at initial state and steady state



Conclusions and further work

In this report, we presented a work to include joint trips simulation in the iterative multi-agent traffic simulation software MATSim. The proposed approach is to optimise plans for groups of agents, which we named cliques, taking into account pre-identified joint trip possibilities.

Event though it was seen that the current implementation is able to improve the score at the clique level, the convergence is still unsatisfying. In fact, if the algorithm is able to identify pertinent joint trips, the increase of the search space size when including joint trips makes the exploration less efficient. Different solutions may be intended to solve this problem, as adapting the parameters of the algorithm to the size of the problem.

Particularly, it was seen that the proposed algorithm has difficulties to handle large cliques. However, in a big group, as the ones used for our tests, the joint structure is quite sparse: Figure 3.13 shows an average of two joint trips per person in large groups. A possibility to reduce the size of the problems to solve is to only replan part of a clique, for example an individual and its potential co-travelers, considering the non-optimised joint trips of co-travelers as external constraints. This approach would allow to take into account cliques of arbitrary size, and would be suitable to work with joint trips in social networks, which inclusion in MATSim is studied (Hackney (2009)).

However, algorithmic difficulties must not hide the main goal of such a work, which is to better predict human behaviour. Thus, we should also go into the behaviour modeling part in greater depth.

Such an extension could be to include household specific processes, as vehicle allocation, into the model. In fact, even though we tried to stay at the general concept of clique, the most straightforward (and probably the most relevant) kind of co-travelers group is constituted by households.

Moreover, techniques are now available to construct a synthetic population with distribution of both individuals and households being consistent with data sources. Currently, initial plans are affected to individuals, and are assigned based on the individual's characteristics (Ciari *et al.* (2007)). However, evidences show that plans are correlated within an household: for example, so-called "maintenance activities", as shopping, are likely to be allocated to only one member of the household, if any. Thus, generating plans at the household level, within which possible joint trips could be identified, would be a further step towards an household level simulation. Such models are already developed by several researchers, as seen in section 1.2.

Finally, the MATSim software uses iterations between a traffic flow simulation and plan improvement strategies. The traffic flow simulation is a critical part, as it is where the complex interactions between agents are taken into account. However, it is currently impossible to simulate consistently joint trips, where agents have to wait for each other. Including such a capability would be a important improvement.

Bibliography

Axhausen, K. W. (2008) Definition of movement and activity for transport modelling, in D. A. Hensher and K. J. Button (eds.) *Handbook of Transport Modelling*, Elsevier, Oxford.

Baldacci, R., V. Maniezzo and A. Mingozzi (2004) An Exact Method for the Car Pooling Problem Based on Lagrangean Column Generation, *Operations Research*, **52** (3) 422–439.

Balmer, M., K. W. Axhausen and K. Nagel (2006) Agent-Based Demand-Modeling Framework for Large-Scale Microsimulations, *Transportation Research Record*, **1985** (1) 125–134.

Balmer, M., K. Meister, M. Rieser, K. Nagel and K. W. Axhausen (2008) Agent-based simulation of travel demand: Structure and computational performance of MATSim-T, paper presented at the *Innovations in Travel Modeling (ITM'08)*, Portland, June 2008.

Ben-Akiva, M. E. and S. R. Lerman (1985) *Discrete choice analysis : theory and application to travel demand*, MIT Press, Cambridge, MA.

Berridge, S. and J. B. Krawczyk (1998) Relaxation Algorithms in Finding Nash Equilibria, *Technical Report*, Victoria University of Wellington, Wellington, New Zealand.

Bradley, M. and P. Vovsha (2005) A model for joint choice of daily activity pattern types of household members, *Transportation*, **32** (5) 545–571.

Buchholz, F. (1997) The Carpool-Problem, *Technical Report*, Institute of Computer Science, University Stuttgart.

Charypar, D., K. W. Axhausen and K. Nagel (2006) Implementing Activity-Based Models: Accelerating the Replanning Process of Agents Using an Evolution Strategy, paper presented at the *International Conference on Travel Behaviour Research*, Kyoto, August 2006.

Charypar, D. and K. Nagel (2005) Generating complete all-day activity plans with genetic algorithms, *Transportation*, **32**, 369–397.

Ciari, F., M. Balmer and K. W. Axhausen (2007) Mobility tool ownership and mode choice decision processes in multi-agent transportation simulation, paper presented at the *7th Swiss Transport Research Conference*, Ascona, September 2007.

Ciari, F., M. Balmer and K. W. Axhausen (2008) A new mode choice model for a multi-agent transport simulation, paper presented at the *8th Swiss Transport Research Conference*, September 2008.

Eiben, A., R. Hinterding and Z. Michalewicz (1999) Parameter control in evolutionary algorithms, *IEEE Transactions on Evolutionary Computation*, **3** (2) 124–141.

Fang, Z., W. Tu, Q. Li and Q. Li (2011) A multi-objective approach to scheduling joint participation with variable space and time preferences and opportunities, *Journal of Transport Geography*, **19** (4) 623–634.

Feil, M., M. Balmer and K. W. Axhausen (2009) New approaches to generating comprehensive all-day activity-travel schedules, *Research Report*, **575**, IVT, ETH Zurich, Zurich.

Flötteröd, G., M. Bierlaire and K. Nagel (2010a) Bayesian demand calibration for dynamic traffic simulations, *Technical Report*, TRANSP-OR, EPF Lausanne, Lausanne.

Flötteröd, G., Y. Chen and K. Nagel (2010b) Behavioral calibration and analysis of a large-scale travel microsimulation, *Technical Report*, TRANSP-OR, EPF Lausanne, Lausanne.

Gliebe, J. P. and F. S. Koppelman (2002) A model of joint activity participation between household members, *Transportation*, **29** (1) 49–72.

Gliebe, J. P. and F. S. Koppelman (2005) Modeling household activity–travel interactions as parallel constrained choices, *Transportation*, **32** (5) 449–471.

Golob, T. (2000) A simultaneous model of household activity participation and trip chain generation, *Transportation Research Part B: Methodological*, **34** (5) 355–376.

Golob, T. and M. McNally (1997) A model of activity participation and travel interactions between household heads, *Transportation Research Part B: Methodological*, **31** (3) 177–194.

Grefenstette, J. (1986) Optimization of control parameters for genetic algorithms, *IEEE Transactions on Sytems Man and Cybernetics*, **16** (1).

Hackney, J. K. (2009) Integration of social networks in a large-scale travel behavior microsimulation, Ph.D. Thesis, ETH Zurich, Zurich.

Harik, G. (1995) Finding multimodal solutions using restricted tournament selection, paper presented at the *Proceedings of the Sixth International Conference on Genetic Algorithms*, 24–31, Pittsburg, PA, June 1995.

Harik, G. and F. Lobo (1999) A parameter-less genetic algorithm, paper presented at the *Proceedings of the Genetic and Evolutionary Computation Conference*, 258–265, Orlando, Florida, USA, July 1999.

Herrera, F., M. Lozano and A. M. Sanchez (2003) A taxonomy for the crossover operator for real-coded genetic algorithms: An experimental study, *International Journal of Intelligent Systems*, **18** (3) 309–338.

Holland, J. H. (1975) *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*, University of Michigan Press.

Horn, J. (1997) The nature of niching: genetic algorithms and the evolution of optimal, cooperative populations, Ph.D. Thesis, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA.

Horn, J., N. Nafpliotis and D. Goldberg (1994) A niched Pareto genetic algorithm for multiobjective optimization, paper presented at the *IEEE World Congress on Computational Intelligence*, 82–87, Orlando, Florida, USA, June 1994.

Horni, A., D. M. Scott, M. Balmer and K. W. Axhausen (2008) Location choice modeling for shopping and leisure activities with MATSim: Combining micro-simulation and time geography, paper presented at the *8th Swiss Transport Research Conference*, Ascona, October 2008.

Jones, P. M., M. C. Dix, M. I. Clarke and I. G. Heggie (1983) *Understanding Travel Behaviour*, Gower, Aldershot.

Koch, R. and M. Skutella (2009) Nash equilibria and the price of anarchy for flows over time, in M. Mavronicolas and V. Papadopoulou (eds.) *Algorithmic Game Theory*, vol. 5814 of *Lecture Notes in Computer Science*, 323–334, Springer.

McNally, M. G. (2000) The activity-based approach, in D. A. Hensher, K. J. Button and . (eds.) *Handbook of transport modelling*, 53–69, Elsevier, Oxford.

Meister, K., M. Balmer and K. W. Axhausen (2005a) An improved replanning module for agent based micro simulations of travel behavior, *Technical Report*, **303**, IVT, ETH Zurich, Zurich.

Meister, K., M. Balmer, K. W. Axhausen and K. Nagel (2006) Planomat: A comprehensive scheduler for a large-scale multi-agent transportation simulation, paper presented at the *6th Swiss Transportation Research Conference*, Ascona, March 2006.

Meister, K., M. Balmer, F. Ciari, A. Horni, M. Rieser, R. A. Waraich and K. W. Axhausen (2010) Large-scale agent-based travel demand optimization applied to switzerland, including mode choice, paper presented at the *12th WCTR*, Lisbon, July 2010.

Meister, K., M. Frick and K. W. Axhausen (2005b) A GA-based household scheduler, *Transportation*, **32** (1) 473–494.

Michalewicz, Z. and C. Janikow (1991) Handling constraints in genetic algorithms, paper presented at the *Proceedings of the Fourth International Conference on Genetic Algorithms*, 151–157, San Diego, CA, USA, July 1991.

Michalewicz, Z. and C. Janikow (1996) GENOCOP: a genetic algorithm for numerical optimization problems with linear constraints, *Communications of the ACM*, **39** (12).

Miller, E. J., M. J. Roorda and J. A. Carrasco (2005) A tour-based model of travel mode choice, *Transportation*, **32** (4) 399–422.

Nisan, N., T. Roughgarden, E. Tardos and V. V. Vazirani (eds.) (2007) *Algorithmic game theory*, Cambridge University Press, July 2007.

Recker, W. (1986) A model of complex travel behavior: Part I—Theoretical development, *Transportation Research Part A: General*, **20** (4) 307–318.

Recker, W. (1995) The household activity pattern problem: General formulation and solution, *Transportation Research Part B: Methodological*, **29** (1) 61–77.

Rieser, M., K. Nagel, U. Beuck, M. Balmer and J. Rümenapp (2007) Truly agent-oriented coupling of an activity-based demand generation with a multi-agent traffic simulation, *Transportation Research Record*, **2021**, 10–17.

Van Veldhuizen, D. A. and G. B. Lamont (2000) Multiobjective evolutionary algorithms: analyzing the state-of-the-art., *Evolutionary computation*, **8** (2) 125–47.

Zhang, J., H. Timmermans and A. Borgers (2005) A model of household task allocation and time use, *Transportation Research Part B: Methodological*, **39** (1) 81–95.

Appendix

.1 Short presentation of the Institute for Transport Planning and Systems (IVT)

This project was realised at the Institute for Transport Planning and Systems (Institut für Verkehrsplanung und Transportsysteme, IVT) of the Federal Institute of Technology of Zürich, Switzerland (ETHZ). It is part of the Department of Civil, Environmental and Geomatic Engineering.

The institute is divided into three groups:

1. Road Traffic Technique, focusing operational aspects of traffic flows.
2. Transport Systems, focusing with the operational aspects of public transport.
3. Transport Planning, where this project was done, focusing on modeling and planing of transport systems. An large portion of the members of this group work with the multi-agent traffic simulation software MATSim, which is developed by members of the IVT, of the VPL laboratory in TU Berlin, as well as by the Senozon young company.