CS524 Project:
StarCraft Mixed Integer Program

Fall 2016
David Merrell
dmerrell@cs.wisc.edu

I. Background

StarCraft is a real-time strategy PC game released by Blizzard Entertainment in 1998. It is widely regarded as the defining game of the genre, receiving broad critical acclaim and many years of commercial success. The game retains a large and loyal fanbase and is a popular e-sport, especially in Korea. StarCraft also contributes to artificial intelligence research, serving as a testbed for agent design competitions.

The objective of a StarCraft match is to defeat one's opponent militarily by destroying all of their assets. Successful play entails gathering resources, building a base, building an army, and leading it to victory. Military strategy and tactics are necessary to win, but arguably the player's most important decisions concern management of resources and production.

Generally speaking, real-time strategy (RTS) games provide rich decision spaces that share many commonalities with real-world decision making—resource constraints, scheduling of interdependent tasks, competing priorities, and uncertainty. While an RTS game is necessarily simpler than the real world, it provides a level of abstraction that real-world problems might be reduced to. Insights might be shared between them.

II. Problem Description

The first minutes of a StarCraft match frequently decide its outcome. During this time, players make crucial decisions about resource management and production in order to build their bases and militaries as quickly as possible. Players who grow the fastest are at an advantage; in some cases they are able to overwhelm their opponents within minutes of the game's start. This is a strategy known as "rushing".

In this research, we built an optimization model to answer the following question: *What choices in the first minutes of a StarCraft match best prepare a player for a rush by their opponent?*

Restated more clearly as an optimization problem, we ask *what production and allocation choices maximize a player's combat strength within the first minutes of a StarCraft match, subject to constraints imposed by game mechanics?*

We chose to restrict the scope of our research to gameplay prior to any encounter with the opponent; we are modeling growth and production, not combat. Furthermore, while StarCraft has three distinct teams or "races" that a player might choose to be (humans or various kinds of extraterrestrials), we currently only model gameplay for the human race (referred to as "terran" within the game).

III. Model Formulation

In order to solve this problem, we formulated a Mixed Integer Program (MIP) optimization model that maximizes a player's combat strength, while obeying the game mechanics of StarCraft. In formulating this model, we had to (a) discretize time, (b) design an objective function, and (c) represent game mechanics using linear constraints. All of the data used in this project was obtained from the "Liquipedia" StarCraft encyclopedia: http://wiki.teamliquid.net/starcraft/Main_Page.

III.a. Discretizing Time. While StarCraft is known as a "Real Time Strategy" game, it is a computer game and operates in discrete steps (referred to as "logical steps" on Liquipedia). At "normal" game speed, there are about 15 of these logical steps per second. Modeling at this timescale would have become unwieldy for any reasonable duration of gameplay; if each timestep required tens of variables and tens of constraints, then a model for five minutes of gameplay would involve tens of thousands of variables and constraints.

Fortunately, almost all of the relevant times (e.g. the time required to build each entity) were multiples of 30 logical steps, meaning that our model could use timesteps of approximately 2 seconds without incurring undue discretization error. The only quantities that didn't fit cleanly into the 2-second discretization were the durations of resource-gathering sorties; in section IV, we address this shortcoming with a sensitivity analysis.

III.b. Designing the Objective. Determining a sensible objective function was one challenging aspect of formulating this model. There is not one obviously correct way to measure a player's combat strength. In a sophisticated game like StarCraft, combat strength is nonlinear due to the complementarity of different types of entities. However, for the purposes of this model we assumed combat strength to be the sum of individual entities' strengths; entity strength was estimated with the following metric:

$$S_e = f_e d_e \left(h_e + r_e \alpha\right). \tag{1}$$

Where f_e, d_e, h_e, and r_e are the entity's attack rate, attack damage, health points, and attack range respectively; and α is a weight describing the advantage given by attack range. This entity strength metric arises from considering a simplified one-on-one confrontation between two entities, A and B. One finds that a necessary condition for A to defeat B is this inequality:

$$f_A d_A \left(h_A + r_A \frac{f_B d_B}{v_A}\right) \geq f_B d_B \left(h_B + r_B \frac{f_A d_A}{v_A}\right),$$

suggesting that entities might be usefully compared by the quantity given in (1). We found that setting $\alpha = 0.5$ yielded reasonable comparative strengths.

Finally, due to uncertainty imposed by the opponent, we let the objective be the sum of the strengths of existing entities over all timesteps past the two-minute mark. In effect, we assume that the opponent may attack with equal likelihood at any point after two minutes of play. In summary, our model seeks to maximize

$$\sum_{t \geq 60}^{150} \sum_{e \in E} S_e \cdot Q_{t,e} \tag{2}$$

where $Q_{t,e}$ is the quantity of entity type e that exists at timestep t, and S_e is the strength of entity type e, as given by (1).

III.c. Formulating the Constraints. Decision making in RTS games is typically subject to a variety of resource, dependency, and availability constraints. StarCraft is sophisticated in this respect; production is constrained by two gathered resources ("minerals" and "gas"), and a derived resource ("supply"). Furthermore, StarCraft entities exist on a tree of dependencies; certain things must be made before other things *can* be made. Each action takes time, and typically requires availability of labor.

In the end, all of the relevant game mechanics are expressed as linear constraints in our MIP. We show some of them in this section; the remainder can be found in the `sc-mip-model.gms` file.

- **Continuous Mineral Update.** In a preliminary version of the model, we simplified resource gathering by assuming resources accumulated continuously, at a rate proportional to the number of workers

allocated to them. We would later replace this with a discrete mineral update model (shown next). In section IV, we see that this makes a substantial difference.

$$\text{minerals}_t = \text{minerals}_{t-1} + \text{mineralWorkers}_{t-1} \cdot \frac{\text{(load size)}}{\text{(sortieTime)}} - \sum_{e \in E} \text{buildEntity}_{t-1,e} \cdot \text{mineralCost}_e$$

- **Discrete Mineral Update.** Minerals are gathered in discrete loads, and spent on entity production. Mineral gathering sorties are rounded down to 5 timesteps. In section IV, we observe only small differences in the results by rounding up to 6 timesteps.

$$\text{minerals}_t = \text{minerals}_{t-1} + \text{(load size)}(\text{workersBeginSortie}_{t-5}) - \sum_{e \in E} \text{buildEntity}_{t-1,e} \cdot \text{mineralCost}_e$$

- **Technical Dependency Tree.** Production of entity type j is dependent on the existence of entity type i for certain pairs $(i, j) \in E^2$. M_j is an upper limit to the number of entity type j that might be built in a timestep.

$$\text{buildEntity}_{t,j} \leq M_j \cdot Q_{t,i}$$

- **Various Worker Constraints.** Workers gather resources and build buildings. Modeling the availability of their labor required a variety of constraints and auxiliary variables. Note that $B \subset E$ is the set of building types.

$$Q_{t,\text{worker}} \geq \text{mineralWorkers}_t + \text{gasWorkers}_t + \text{buildingWorkers}_t;$$

$$\text{buildingWorkers}_t \geq \sum_{b \in B} \left(Q_{t+\text{buildTime},b} - Q_{t,b} \right);$$

$$\text{mineralWorkers}_t \geq \sum_{t'=t-5}^{t} \text{workersBeginSortie}_{t'}.$$

IV. Solves, Results, and Analysis

Four versions of the model were solved to within 0.1% of proven optimality. The first version approximated resource gathering as a continuous process. The second version depicted the discrete nature of mineral gathering; however, it had a weakness in that it rounded the true duration of a mineral-gathering sortie down to 5 timesteps. The third version of the model tested the impact of this rounding down, by instead rounding the mineral-gathering sortie duration up to 6 timesteps. The fourth version included discrete mineral gathering and examined the effect of modifying the objective, such that workers' combat strengths do not contribute to the total combat strength.

All four instances were run within a total of six minutes, using the CPLEX MIP solver on a laptop PC. The continuous mineral gathering model consisted of 6,577 equations with 5,530 variables, and was solved in 140s. The instances that included discrete mineral gathering consisted of 6,727 equations in 5,680 variables; their solve times ranged between 30s and 80s. In each solution, the model did not progress beyond the most basic levels of the dependency tree; this can be attributed to the model's narrow five-minute time frame. The results shown in figure 1 indicate that discretized mineral gathering yields substantially different results from continuous mineral gathering, and that changes to the objective lead to large changes in the solution.

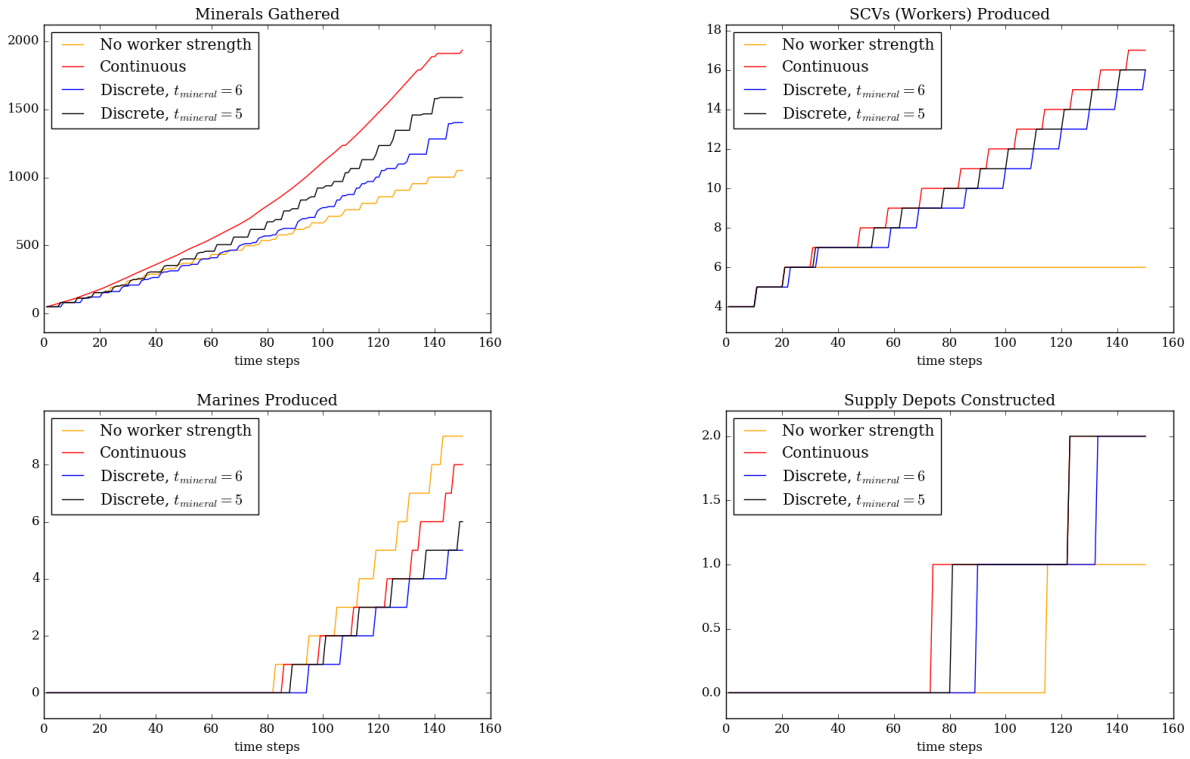

Figure 1: Gathering and production for different versions of the model. "No worker strength" refers to the case where workers' combat strengths do not contribute to the objective.

However, the difference between instances using discrete mineral gathering with 5-timestep sorties and 6-timestep sorties is small over our five-minute timeframe. Qualitatively, the actions taken in the 6-timestep instance are very similar to those in the 5-timestep instance, but occur more slowly.

In reviewing the four solutions, we reject the continuous mineral gathering instance for being overly optimistic; we reject the instance with a modified objective, due to its low production of workers (in real gameplay, having a full complement of workers is very desirable). We take the discrete gathering instance with 5-second sortie durations as the most viable course of action—a course of action that best prepares the player for early confrontations with the opponent. It is displayed graphically below, with time expressed in seconds of "normal" game speed.